

Компьютерные системы и сети

Выпуск 3

Компьютерные системы и сети

Серия основана в 2013 году

Ответственный редактор А.В. Пролетарский

РЕДАКЦИОННЫЙ СОВЕТ:

А.А. Александров (*председатель*), д-р техн. наук
М.А. Басараб, д-р физ.-мат. наук
В.В. Девятков, д-р техн. наук
И.П. Иванов, д-р техн. наук
А.П. Карпенко, д-р техн. наук
Е.А. Микрин, академик РАН
А.В. Пролетарский, д-р техн. наук
И.В. Рудаков, канд. техн. наук
В.В. Сюзев, д-р техн. наук
В.М. Черненький, д-р техн. наук
А.Б. Шаповалов, д-р техн. наук
В.А. Шахнов, член-корр. РАН

Москва
Издательство МГТУ им. Н.Э. Баумана
2019

Компьютерные системы и сети

Выпуск 3

Е.В. Смирнова, А.В. Пролетарский, Е.А. Ромашкина

Технологии TCP/IP в современных компьютерных сетях

Допущено Федеральным учебно-методическим объединением в системе высшего образования по укрупненной группе специальностей и направлений подготовки 09.00.00 «Информатика и вычислительная техника» в качестве учебного пособия для студентов (адъюнктов), обучающихся по основным образовательным программам высшего образования по направлениям подготовки бакалавриата/магистратуры укрупненной группы специальностей и направлений подготовки 09.00.00 «Информатика и вычислительная техника»



BAUMANPRESS
МОСКВА
ИЗДАТЕЛЬСТВО
МГТУ им. Н. Э. БАУМАНА
2019

УДК 004.7
ББК 32.973.202
С50

Р е ц е н з е н т ы:

генеральный директор АО «РтСофт», д-р техн. наук *О.В. Синенко*;
директор фирмы «1С», канд. экон. наук *Б.Г. Нуралиев*

Смирнова, Е.В.

Технологии TCP/IP в современных компьютерных сетях : учебное пособие /
C50 Е. В. Смирнова, А. В. Пролетарский, Е. А. Ромашкина . — Москва : Издательство
МГТУ им. Н.Э. Баумана, 2019. — 436, [4] с. : ил. — (Компьютерные системы и сети).

ISBN 978-5-7038-5166-1

Книга посвящена изучению стека протоколов TCP/IP - технологической основе сети интернет. Описывается стек протоколов TCP/IP. Рассматривается протокол PPP, функционирование канала PPP, LCP, NCP, протоколы аутентификации PPP, сжатие данных, протоколы шифрования, протоколы туннелирования, передача PPP через Ethernet и ATM, протокол PPTP, протокол L2TP, типы подключения к провайдерам. Изучается протокол IP версии 4 и версии 6, архитектура безопасности. Рассматриваются протоколы разрешения адресов ARP, Gratuitous ARP, Proxy ARP, разрешение адресов для IPv6. Представлен протокол ICMP, протокол NDP. Разрешение адресов IPv6 и определение недоступности соседа, определение дублирования адресов, обнаружение маршрутизатора. Изучается технология маршрутизации, IP-интерфейсы маршрутизирующих коммутаторов, архитектура протоколов маршрутизации, алгоритмы маршрутизации, дистанционно-векторные протоколы маршрутизации, протокол OSPF, включая версию 3. Рассматриваются протоколы транспортного уровня. Адресация протоколов TCP и UDP, протокол UDP, протокол TCP. Представлены протоколы уровня приложений Telnet, SSH, SSL/TLS, DHCP, DHCPv6. Изложена методика поиска неисправностей в сетях TCP/IP. Предлагаемые практические работы охватывают все рассмотренные темы. Издание содержит обширный глоссарий.

Учебное пособие является результатом многолетнего сотрудничества МГТУ им. Н.Э. Баумана и компании D-Link по подготовке кадров для сферы информационно-коммуникационных технологий.

УДК 004.7
ББК 32.973.202

ISBN 978-5-7038-5166-1

© Смирнова Е.В., Пролетарский А.В., Ромашкина Е.А., 2019
© Оформление. Издательство
МГТУ им. Н.Э. Баумана, 2019

Оглавление

Предисловие	8
1. Обзор TCP/IP	10
1.1. История TCP/IP	10
1.2. Стек протоколов TCP/IP.....	11
2. Протокол PPP	13
2.1. Общий формат кадра PPP.....	14
2.2. Функционирование канала PPP	15
2.3. Link Control Protocol (LCP)	17
2.4. Network Control Protocol (NCP).....	21
2.5. Протоколы аутентификации PPP	23
2.5.1. Протокол Password Authentication Protocol (PAP)	24
2.5.2. Протокол Challenge Handshake Authentication Protocol (CHAP).....	26
2.5.3. Протокол Microsoft Challenge Handshake Authentication Protocol (MS-CHAP).....	29
2.6. Сжатие данных в PPP	32
2.7. Протоколы шифрования данных PPP	33
2.7.1. Протокол шифрования MPPE	34
2.8. Протоколы туннелирования PPP	37
2.9. Передача PPP через Ethernet	38
2.9.1. Функционирование PPPoE.....	42
2.10. Передача PPP через ATM	48
2.10.1. Обзор технологии ADSL.....	49
2.10.2. Обзор технологии ATM.....	56
2.11. Протокол PPTP.....	62
2.12. Протокол L2TP	67
2.13. Типы подключения к провайдерам	78
3. Протокол IP	83
3.1. Протокол IP версии 4	84
3.1.1. Поле Type of Service	86
3.1.2. Фрагментация пакетов IPv4.....	88
3.1.3. Понятие IP-адресации.....	91
3.1.4. Представление и структура адреса IPv4.....	92
3.1.5. Классовая адресация IPv4.....	94
3.1.6. Частные и публичные адреса IPv4	96
3.1.7. Формирование подсетей.....	97
3.1.8. Маски подсети переменной длины (VLSM).....	101
3.1.9. Бесклассовая адресация IPv4.....	103
3.1.10. Технология NAT	108
3.1.11. Многоадресная передача пакетов IPv4	112
3.2. Протокол IP версии 6	115
3.2.1. Формат заголовка IPv6	117
3.2.2. Размер пакета IPv6.....	120
3.2.3. Представление и структура адреса IPv6	121
3.2.4. Типы адресов IPv6	123
3.2.5. Индивидуальные адреса	124
3.2.6. Альтернативные адреса	130
3.2.7. Групповые адреса	131

Оглавление

3.2.8. Способы конфигурации адреса IPv6	133
3.2.9. Планирование подсетей IPv6	137
3.3. Обзор архитектуры безопасности для протокола IP	139
3.3.1. Компоненты IPSec	142
3.3.2. Протокол Encapsulating Security Payload (ESP)	152
3.3.3. Протокол Internet Key Exchange (IKE)	158
3.3.4. Использование NAT в протоколе IPSec	167
3.3.5. Определение жизнеспособности IPSec-соединения	172
4. Протоколы разрешения адресов	175
4.1. Протокол ARP	176
4.1.1. Операции ARP	178
4.2. Gratuitous ARP	180
4.3. Proxy ARP	182
4.4. Разрешение адресов для IPv6	182
5. Протокол ICMP	184
5.1. Классы, типы и коды сообщений ICMP	186
5.2. Правила генерации сообщений ICMP	188
5.3. Утилита Ping	189
6. Протокол NDP	192
6.1. Разрешение адресов IPv6 и определение недоступности соседа	193
6.2. Определение дублирования адресов	196
6.3. Обнаружение маршрутизатора	197
7. Понятие маршрутизации	198
7.1. IP-интерфейсы маршрутизирующих коммутаторов	204
7.2. Архитектура протоколов маршрутизации	207
7.3. Алгоритмы маршрутизации	209
7.4. Дистанционно-векторные протоколы маршрутизации	213
7.4.1. Протокол RIP	214
7.4.2. Проблемы при функционировании дистанционно-векторного алгоритма маршрутизации	220
7.4.3. Протокол RIPv2	224
7.4.4. Протокол RIPng	226
7.5. Протокол OSPF	227
7.5.1. Обзор протокола	228
7.5.2. Типы пакетов протокола OSPF	234
7.5.3. Состояния соседства	238
7.5.4. Установление соседства	239
7.5.5. Вычисление маршрутов	249
7.5.6. Обновление маршрутной информации внутри области	251
7.6. Протокол OSPF версии 3	253
7.6.1. Пакеты OSPFv3	255
7.6.2. Обзор LSA OSPFv3	256
8. Протоколы транспортного уровня	259
8.1. Адресация протоколов TCP и UDP	259
8.2. Протокол UDP	264
8.2.1. Форматдейтаграммы UDP	265
8.3. Протокол TCP	265
8.3.1. Сегмент TCP	266
8.3.2. Модель управления TCP-соединением	268

Оглавление

8.3.3. Установка соединения TCP	271
8.3.4. TCP Fast Open	275
8.3.5. Подтверждения и повторная передача.....	280
8.3.6. Завершение соединения TCP.....	281
8.3.7. Механизм скользящего окна	282
8.3.8. Контроль и предотвращение перегрузки в TCP.....	285
8.3.9. Явное уведомление о перегрузке (ECN)	291
8.3.10. Функция Virtual Server.....	294
9. Протоколы уровня приложений.....	297
9.1. Протокол Telnet	297
9.2. Протокол SSH	299
9.2.1. Транспортный протокол SSH.....	300
9.2.2. Протокол аутентификации пользователей SSH.....	307
9.2.3. Протокол соединений SSH	308
9.3. Протоколы SSL/TLS	311
9.3.1. Архитектура SSL/TLS	313
9.3.2. Основные отличия TLS 1.2 и TLS 1.3	314
9.3.3. Протокол Change Cipher Spec	315
9.3.4. Протокол Alert	316
9.3.5. Сертификаты X.509	316
9.3.6. Протокол Handshake в TLS 1.2	321
9.3.7. Протокол Handshake в TLS 1.3	330
9.3.8. Протокол Record	339
9.4. Протокол DHCP	340
9.4.1. Архитектура DHCP	341
9.4.2. Формат сообщения DHCP	345
9.4.3. Взаимодействие между клиентом и сервером DHCP	348
9.4.4. Функционирование relay-агента DHCP.....	357
9.4.5. Опция DHCP Relay Agent Information (Option 82)	366
9.4.6. Функция DHCP Local Relay.....	370
9.4.7. Технология DHCP Snooping.....	371
9.5. Протокол DHCPv6	376
9.5.1. Типы сообщений DHCPv6	378
9.5.2. Уникальный идентификатор DHCP (DUID)	381
9.5.3. Ассоциация идентичности (IA)	383
9.5.4. Stateful DHCPv6	384
9.5.5. Stateless DHCPv6	392
9.5.6. DHCPv6 Prefix Delegation.....	394
9.5.7. Опции DHCPv6 Relay Agent.....	400
9.5.8. Функция DHCPv6 Guard	400
10. Поиск неисправностей в сетях TCP/IP	402
10.1. Методика поиска неисправностей	402
10.2. Средства поиска и устранения неполадок.....	403
10.3. Анализ неисправностей.....	404
10.3.1. Проверка параметров протокола IP	406
10.3.2. Проверка физического соединения	410
10.3.3. Проверка канального уровня.....	416
10.3.4. Проверка сетевого уровня	417
10.3.5. Проверка протоколов верхних уровней.....	420
Глоссарий	427

Предисловие

Развитие технологий в сети интернет определило бурную цифровизацию нашей жизни. Уже обыденностью стал интернет вещей, нас окружает виртуальная и дополненная реальности. Через интернет идет улучшение сферы здравоохранения и повышение качества повседневной жизни, доступнее становится образование, развиваются социальные сети. Все это, и многое другое, стало возможным благодаря технологической основе сети интернет – стеку протоколов TCP/IP.

Наша книга посвящена подробному изучению протоколов TCP/IP – набору правил, определяющему как доставить информацию в сети быстро и безопасно. Ее выпуск стал результатом многолетнего сотрудничества МГТУ им. Н.Э. Баумана и компании D-Link по подготовке кадров для сферы информационно-коммуникационных технологий. Это третья по счету книга в серии «Компьютерные системы и сети» после книг «Технологии коммутации и маршрутизации в локальных компьютерных сетях» и «Технологии современных беспроводных сетей Wi-Fi». Вместе с ранее изданными книгами по IP-телефонии, технологиям защиты информации в компьютерных сетях, управлению коммутируемой средой авторы предоставляют возможность изучить широкий спектр информационно-коммуникационных технологий, повысить квалификацию специалистам.

Учебное пособие состоит из 10 глав, 22 практических работ и обширного глоссария.

Глава 1 посвящена истории TCP/IP, описывается стек протоколов TCP/IP.

В главе 2 рассматривается протокол PPP. Общий формат кадра PPP, функционирование канала PPP, LCP, NCP, протоколы аутентификации PPP, сжатие данных, протоколы шифрования, протоколы туннелирования, передача PPP через Ethernet и ATM, протокол PPTP, протокол L2TP, типы подключения к провайдерам.

Глава 3 посвящена протоколу IP версии 4 и версии 6, архитектуре безопасности.

В главе 4 рассматриваются протоколы разрешения адресов ARP, Gratuitous ARP, Proxy ARP, разрешение адресов для IPv6.

В главе 5 изучается протокол ICMP. Классы, типы и коды сообщений ICMP, правила генерации сообщений ICMP, утилита Ping.

Глава 6 посвящена протоколу NDP. Разрешение адресов IPv6 и определение недоступности соседа, определение дублирования адресов, обнаружение маршрутизатора.

В главе 7 вводится понятие маршрутизации. Изучаются IP-интерфейсы маршрутизирующих коммутаторов, архитектура протоколов маршрутизации, алгоритмы маршрутизации, дистанционно-векторные протоколы маршрутизации, протокол OSPF, включая версию 3.

В главе 8 рассматриваются протоколы транспортного уровня. Адресация протоколов TCP и UDP, протокол UDP, протокол TCP.

Предисловие

Глава 9 посвящена протоколам уровня приложений Telnet, SSH, SSL/TLS, DHCP, DHCPv6.

В главе 10 изложена методика поиска неисправностей в сетях TCP/IP.

Практическая часть содержит 22 лабораторные работы, охватывающие все теоретические вопросы. Теоретический материал и выполненные практические задания дают полный объем знаний и компетенций по технологиям информационного обмена в сети.

Обозначения, используемые в книге

В тексте книги для обозначения сетевых устройств различных типов используются следующие пиктограммы:



Коммутатор



DSLAM



Маршрутизатор/
Коммутатор L3



Беспроводной
маршрутизатор



Рабочая
станция



Портативный
компьютер



Персональный
компьютер



Сервер



Сетевая
среда



Беспроводная
среда



Злонамеренный
пользователь



Шлюз
безопасности

1. Обзор TCP/IP

1.1. История TCP/IP

Термин TCP/IP, который относится к целому семейству протоколов, образован из названий двух из них: Transmission Control Protocol (TCP) и Internet Protocol (IP). Протоколы семейства TCP/IP начали разрабатывать как часть экспериментальной сети ARPAnet, созданной Агентством перспективных исследований Министерства обороны США (United States Defense Advanced Research Projects Agency, DARPA, или ARPA). Первоначально сеть ARPAnet использовала адаптированные к ее требованиям существующие на тот момент протоколы. Однако все они имели какие-либо недостатки или ограничения. Разработчики новой сети поняли, что использование имеющихся протоколов приведет к существенным проблемам по мере ее расширения.

В 1973 году началась разработка полноценной системы протоколов межсетевого обмена для сети ARPAnet. Самая ранняя ее версия, написанная в 1973 году, содержала описание только одного протокола: TCP. Эта аббревиатура означала «Transmission Control Program». Далее эта версия была доработана и в декабре 1974 года формально документирована в RFC 675 «Specification of Internet Transmission Control Program».

Тестирование и исследование TCP продолжались несколько лет. В марте 1977 года была документирована вторая версия TCP. В августе 1977 года произошел переломный момент в разработке TCP/IP. Джон Постел (Jon Postel), являющийся одним из разработчиков TCP/IP и Интернета, опубликовал в Internet Engineering Note number 2 ряд комментариев о состоянии TCP. В частности, он отметил, что новый протокол пытается выполнять слишком много функций и должен использовать принцип разбиения на уровни. Это замечание Постела привело к созданию архитектуры TCP/IP и разбиению первоначального TCP (Transmission Control Program) на два уровня: Transmission Control Protocol (TCP) на транспортном уровне и Internet Protocol (IP) на сетевом уровне. Процесс разбиения был описан в 1978 году в третьей версии TCP. Первая версия стандартов TCP и IP, используемая в современных сетях, была документирована в 1980 году как TCP version 4 и IP version 4. По этой причине у протокола IP первая версия 4, а не 1. TCP/IP быстро стал набором протоколов для ARPAnet, а позже, в 1983 году — для Интернета.

Успех стека протоколов TCP/IP определяется как историческими факторами (протоколы для Интернета), так и техническими характеристиками, включающими интегрированную адресную систему, возможность маршрутизации, независимость от нижележащих технологий LAN, WLAN и WAN, масштабируемость, использование открытых стандартов и универсальность.

1.2. Стек протоколов TCP/IP

Стек протоколов TCP/IP был создан раньше модели OSI, поэтому его разработчики не использовали модель OSI для описания архитектуры стека. Они разработали собственную модель, которая имела несколько названий, включая *модель TCP/IP* (Transmission Control Protocol/Internet Protocol), *модель DARPA* (Defense Advanced Research Projects Agency (DARPA, или ARPA)) либо *модель DOD* (United States Department of Defense).

Так как модель OSI имеет широкое распространение, архитектура TCP/IP часто описывается с использованием названий уровней модели TCP/IP и соответствующих уровней модели OSI.

Модель TCP/IP, так же как и модель OSI, имеет многоуровневую структуру, но для того чтобы данные от приложения компьютера А были переданы приложению на компьютере В, они должны последовательно пройти четыре уровня: уровень приложений, транспортный уровень, уровень Интернета и уровень доступа к среде.

Как показано на рис. 1.1, трем верхним уровням в модели OSI соответствует **уровень приложений** (*Application layer*) в модели TCP/IP, который включает в себя функции представления, кодирования и контроля над установлением соединения. Существует множество протоколов уровня приложений, из которых самыми распространенными являются FTP, TFTP, HTTP/HTTPPs, DHCP, DNS, Telnet, SMTP, POP3, IMAP и др.

Транспортный уровень (*Transport layer*) модели TCP/IP (рис. 1.2) выполняет те же функции, что и одноименный уровень в модели OSI. На этом уровне определены два протокола — TCP и UDP. Протокол TCP (Transmission Control Protocol) обеспечивает надежную доставку сегментов по сети за счет установления логического соединения между отправителем и получателем данных. Протокол UDP (User Datagram Protocol), в отличие от TCP,

Модель OSI	Модель TCP/IP
Уровень приложений	Уровень приложений (Application)
Уровень представлений	
Сеансовый уровень	
Транспортный уровень	Транспортный уровень (Transport)
Сетевой уровень	Уровень Интернета (Internet)
Канальный уровень	
Физический уровень	Уровень доступа к среде (Network Access)

Рис. 1.1. Соответствие между уровнями в модели OSI и в модели TCP/IP

Технологии TCP/IP в современных компьютерных сетях

не устанавливает соединение между отправителем и получателем сообщения и не гарантирует надежную доставку данных.

Уровень Интернета (Internet layer) аналогичен по функциям сетевому уровню модели OSI и обеспечивает организацию связи между сетями и подсетями, образующими составную сеть. Основным протоколом уровня Интернета является IP, который выполняет две основные функции — адресацию узлов и выбор наилучшего маршрута до сети назначения (маршрутизацию). Также на этом уровне работают протоколы ICMP, IGMP, протоколы маршрутизации RIP, OSPF, BGP.

Уровень доступа к среде (Network access layer) объединяет функции канального и физического уровня модели OSI, обеспечивая физическую передачу данных в сети. Существует множество различных протоколов уровня доступа к сети, из которых самыми распространенными являются Ethernet, IEEE 802.11 (Wi-Fi), PPP, ATM и др.

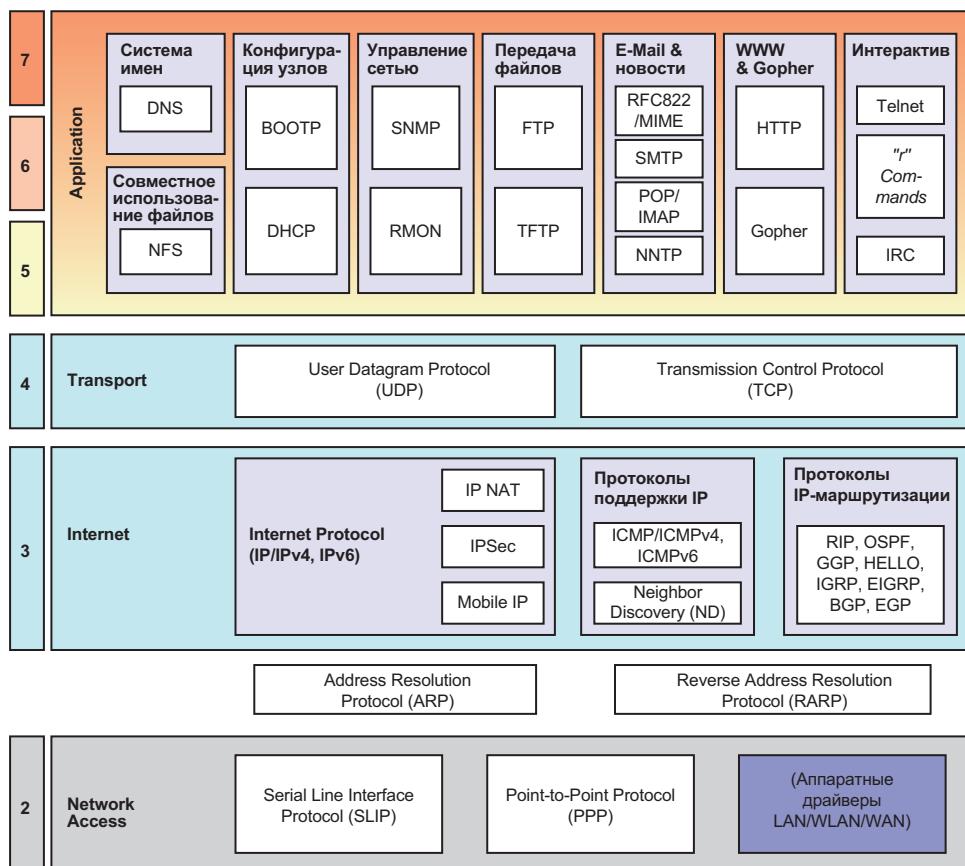


Рис. 1.2. Протоколы стека TCP/IP

2. Протокол PPP

Стек протоколов TCP/IP подразумевает, что функциональность второго уровня обеспечивается технологиями локальных и глобальных сетей. В первой и второй частях курса были подробно рассмотрены технологии локальных сетей Ethernet и 802.11. При подключении в Интернета зачастую используются двухточечные линии связи, т. е. линии связи, соединяющие устройство клиента с устройством провайдера услуг. Способы подключения локальных сетей к сетям провайдеров могут быть разными: через телефонную линию, по оптоволоконному кабелю, с использованием сотовой связи, кабельных модемов. Одним из протоколов для установления WAN-соединения является протокол PPP (Point-to-Point Protocol, протокол двухточечного соединения). Протокол PPP определяет метод транспортировки дейтаграмм различных протоколов сетевого уровня по последовательным каналам типа «точка-точка». Он описан в RFC 1661 и доработан в более поздних документах RFC 1662 и др. Для работы протокола PPP не требуется никакие дополнительные технологии канального уровня. Он функционирует непосредственно поверх физического соединения, которое использует одну из технологий доступа типа «точка-точка»: Dial-Up, ISDN, ADSL, GPON и т. п.

Существуют расширения протокола PPP, позволяющие передавать пакеты PPP через сети Ethernet (PPPoE) и ATM (PPPoA). Протокол PPP over Ethernet (PPPoE) описан в RFC 2516. Он служит для подключения множества устройств одной сети через единственное абонентское устройство к удаленному концентратору доступа, расположенному на стороне провайдера. Протокол PPPoE используется при подключении в Интернете по технологиям xDSL, ETTx и FTTx. Спецификация PPP over AAL5 (PPPoA) определена в RFC 2364 и описывает инкапсуляцию протокола PPP посредством ATM Adaptation Layer 5 (AAL5). Протокол PPPoA в основном применяется при подключении в Интернете с использованием кабельных модемов (стандарт DOCSIS) и технологий семейства xDSL.

PPP является протоколом с установлением соединения, который позволяет создавать L2-каналы поверх различных соединений физического уровня. Он поддерживает как синхронные, так и асинхронные каналы. Эти каналы, работающие в полнодуплексном режиме, подразумевают, что кадры отправляются и получаются в одном и том же порядке.

Несмотря на то что PPP расшифровывается как Point-to-Point Protocol, его правильнее рассматривать не как протокол, а как стек протоколов. PPP состоит из трех основных компонентов (рис. 2.1):

1) метода для инкапсуляции дейтаграмм множества протоколов. PPP определяет специальный формат кадра для инкапсуляции данных, основанный на кадре, используемом в протоколе HDLC (High-Level Data Link Control).

2) протокола управления каналом (Link Control Protocol, LCP), позволяющего автоматически устанавливать каналы связи, тестировать их,

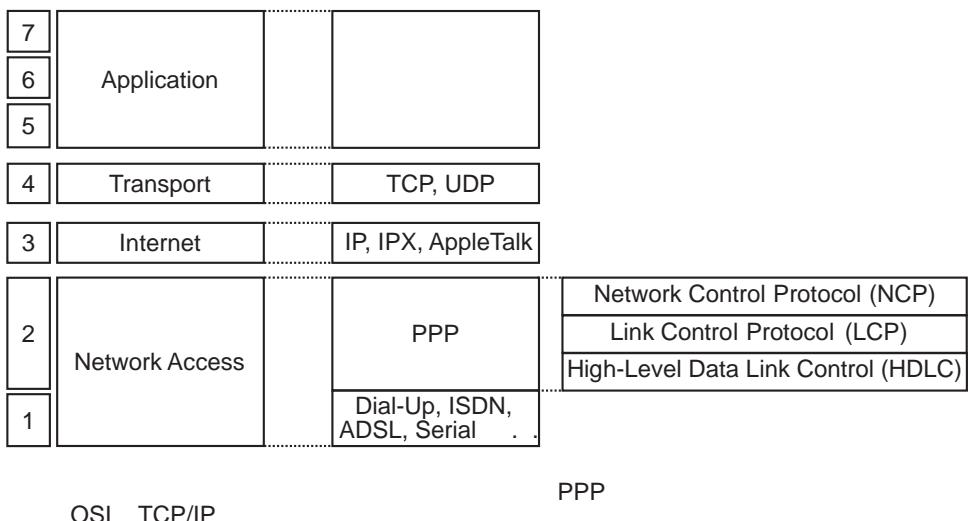


Рис. 2.1. Архитектура PPP

договариваться об их конфигурации и снова отключать, когда они не нужны. Дополнительной возможностью LCP является выполнение аутентификации узлов;

3) семейства протоколов управления сетью (Network Control Protocol, NCP) для установления и конфигурирования различных протоколов сетевого уровня. Для каждого поддерживаемого протокола сетевого уровня должен иметься свой протокол NCP.

2.1. Общий формат кадра PPP

Формат кадра PPP основан на формате кадра HDLC (High-level Data Link Control, высокоуровневый протокол управления каналом). В отличие от бит-ориентированного протокола HDLC, PPP является байт-ориентированным. Структура общего формата кадра PPP описана в RFC 1662 и показана на рис. 2.2.



Рис. 2.2. Общий формат кадра PPP

Все кадры PPP начинаются и заканчиваются флагами длиной 1 байт, значение которых равно 01111110 (0x7E). Они используются для синхронизации кадра. Поле *Адрес* (Address) всегда имеет двоичное значение 11111111 (0xFF). Оно означает, что все станции должны принимать этот кадр.

2. Протокол PPP

Использование такого адреса позволяет избежать назначения станциям индивидуальных адресов, так как PPP работает на канале, связывающем только два устройства. За полем адреса следует поле *Управление* (Control), его значение по умолчанию равно 00000011 (0x03). Это число означает ненумерованный кадр.

Четвертое поле кадра PPP — *Протокол* (Protocol). Его размер составляет 1 или 2 байта, а значение определяет тип протокола, данные которого помещены в поле *Информация*. Поле *Информация* (Information) содержит данные протокола, указанного в поле *Протокол*. При необходимости данные могут дополняться специальными символами в поле *Заполнение* (Padding). Поле *Информация* переменной длины. Оно может содержать как 0, так и большее число байтов. Его максимальная длина называется *Maximum Receive Unit* (MRU) и по умолчанию равна 1500 байт. В процессе установки соединения с помощью LCP стороны могут договориться о значении MRU.

Следом за полем *Информация* располагается поле *Контрольная сумма* (Checksum), длина которого по умолчанию равна 2 байтам. Оно является стандартным 16-битным кодом CRC. В случае необходимости взаимодействующие стороны могут договориться об использовании 4-байтной контрольной суммы, которая представляет собой 32-битный код CRC.

2.2. Функционирование канала PPP

Прежде чем начнется обмен данными по соединению PPP, между двумя устройствами должен быть установлен канал. Для конфигурирования параметров и тестирования канала используется протокол LCP. Конфигурирование параметров выполняется с помощью переговоров. Для этого каждое конечное устройство канала отправляют LCP-пакеты, в которых описывает свои возможности и требования. После установления канала узлы могут выполнить аутентификацию друг друга, если они об этом договорились ранее.

Для выбора и конфигурирования протоколов сетевого уровня, пакеты которых будут передаваться через канал, используются протоколы NCP. Протоколы NCP сложно описать общими словами, так как каждый из них обладает специфическими свойствами, зависящими от соответствующего протокола сетевого уровня, и поддерживает конфигурационные запросы, характерные только для этого протокола. После установления канала PPP отправляет пакеты NCP для конфигурирования поддерживаемых узлами протоколов сетевого уровня. Как только протокол сетевого уровня сконфигурирован, по каналу могут передаваться его дейтаграммы.

Канал будет оставаться открытый до тех пор, пока явно не будет закрыт LCP- или NCP-пакетами или пока не произойдет какое-нибудь внешнее событие.

Диаграмма состояний

Процесс установления, использования и разъединения канала PPP проходит через несколько состояний, показанных на рис. 2.3.

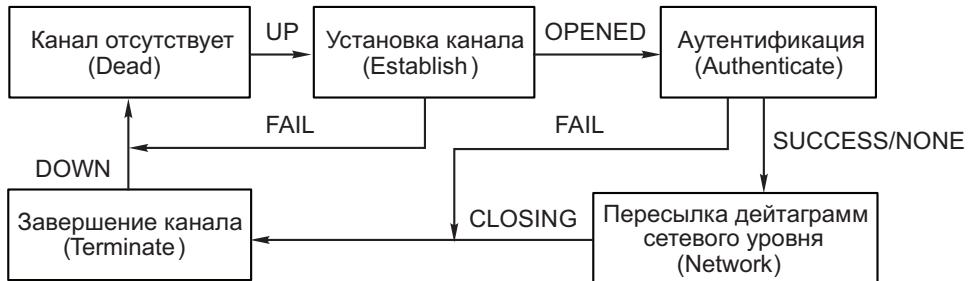


Рис. 2.3. Диаграмма состояний канала PPP

Начальное состояние канала таково: канал отключен (*DEAD*), т. е. соединение на физическом уровне отсутствует. После того как физическое соединение установлено, канал переходит в состояние *ESTABLISH* (установка). В этот момент начинаются переговоры о параметрах с помощью протокола LCP. Узлы PPP обмениваются пакетами LCP для выбора параметров соединения. Инициирующий узел предлагает варианты, а отвечающий узел либо соглашается с ним, либо отвергает частично или полностью. Отвечающий узел также может делать свои предложения. Фаза установки канала завершится, как только протокол LCP перейдет в открытое состояние (*OPENED*), т. е. узлы согласуют конфигурационные параметры.

Прежде чем перейти к обмену пакетами протокола NCP, некоторые узлы могут выполнить аутентификацию. По умолчанию аутентификация не обязательна. Однако если в процессе установки канала с помощью LCP стороны договорились о необходимости выполнения аутентификации, то канал переходит в состояние *AUTHENTICATE* (аутентифицировать). В процессе этой фазы выполняется протокол аутентификации. В случае неуспешной аутентификации канал переходит в состояние *TERMINATE* (завершение). После успешной аутентификации канал переходит в состояние *NETWORK* (сеть). В этом состоянии происходит обмен пакетами NCP для настройки сетевого уровня. После того как NCP перейдет в открытое состояние, PPP начинает передачу пакетов соответствующего протокола сетевого уровня. В данном состоянии трафик на канальном уровне состоит из комбинации пакетов LCP, NCP и протокола сетевого уровня.

Когда передача данных закончена, канал переходит к фазе *TERMINATE* (завершение), а затем возвращается в состояние *DEAD* (отключен), когда физическое соединение разрывается.

Канал PPP может быть закрыт в любое время по требованию пользователя или по другим причинам: из-за потери несущей, неудачной аутентификации, недостаточного качества канала, по истечении допустимого периода времени неиспользования канала.

Для закрытия канала используется протокол LCP. Он выполняет это путем обмена сообщениями *Terminate-Request* и *Terminate-Ack* между взаимодействующими узлами. Когда канал закрывается, PPP информирует об этом

2. Протокол PPP

протоколы сетевого уровня, таким образом, они могут выполнить соответствующие действия.

Закрытие канала с помощью LCP является достаточным. Отправлять поток пакетов *Terminate* каждому протоколу NCP нет необходимости. Более того, тот факт, что один из NCP закрылся, не является достаточной причиной для закрытия канала PPP, даже если этот NCP в тот момент был в открытом состоянии единственным.

2.3. Link Control Protocol (LCP)

Протокол LCP выполняет три важные функции:

- Конфигурирование канала: процесс установления и настройки параметров канала.
- Обслуживание канала: процесс управления и диагностики открытого канала.
- Завершение канала: процесс закрытия существующего канала, если он больше не требуется или если разрывается физическое соединение.

Каждая из этих функций соответствует одному из состояний соединения PPP. Конфигурирование канала выполняется в состоянии *ESTABLISH*. Обслуживание канала выполняется, когда LCP переходит в открытое состояние. Завершение канала происходит в состоянии *TERMINATE*.

Протокол LCP выполняет свои функции путем обмена сообщениями между взаимодействующими устройствами. Эти сообщения называются пакетами LCP. Пакет LCP инкапсулируется в поле *Информация кадра PPP*, в поле *Протокол* которого установлено значение 0xC021.

Общий формат пакета LCP показан на рис. 2.4.

Байты	1	1	2	Переменная
	Код	Идентификатор	Длина	Данные

Значения поля «Код»:

- 1 Configure-Request
- 2 Configure-Ack
- 3 Configure-Nak
- 4 Configure-Reject
- 5 Terminate-Request
- 6 Terminate-Ack
- 7 Code-Reject
- 8 Protocol-Reject
- 9 Echo-Request
- 10 Echo-Reply
- 11 Discard-Request

Рис. 2.4. Общий формат пакета LCP

Поле *Код* (Code) длиной 1 байт описывает тип пакетов LCP. Однобайтное поле *Идентификатор* (Identifier) позволяет определять пары запросов и ответов. Поле *Длина* (Length) имеет длину 2 байта и содержит длину пакета LCP с учетом полей *Код*, *Идентификатор*, *Длина* и *Данные*. Поле *Данные* (Data) переменной длины. Его формат определяется типом пакета LCP, указанным в поле *Код*.

Установление канала

Установление и конфигурирование канала выполняются автоматически путем обмена взаимодействующими узлами конфигурационными пакетами. Узел-инициатор отправляет пакет *Configure-Request*. Он может содержать переменное количество параметров настройки (опций), о которых инициатор желает договориться.

Протоколом LCP согласуются только те опции, которые не зависят от конкретных протоколов сетевого уровня. В RFC 1661 определены шесть конфигурационных опций, которые инициатор может указать в запросе *Configure-Request* (рис. 2.5–2.7):

- *Maximum-Receive-Unit (MRU)*: позволяет указывать максимальный размер передаваемой по каналу дейтаграммы.
- *Authentication Protocol*: позволяет указывать тип протокола аутентификации, если обе стороны желают его использовать. По умолчанию аутентификация не требуется.
- *Quality Protocol*: позволяет указывать тип протокола оценки качества канала. По умолчанию оценка качества канала отключена.
- *Magic Number*: позволяет определять закольцованные каналы или другие нарушения работы канального уровня.
- *Protocol Field Compression*: позволяет договориться о сжатии поля *Протокол* кадра PPP.
- *Address and Control Field Compression (ACFC)*: позволяет договориться о сжатии полей *Адрес* и *Управление* кадра PPP.

Если отвечающий узел соглашается со всеми указанными инициатором параметрами, он отправляет пакет *Configure-Ack*. Как только пакет *Configure-Ack* получен инициатором, протокол LCP переходит в открытое состояние и фаза установления канала завершается.

Если отвечающий узел согласен не со всеми предлагаемыми параметрами, он отправляет узлу-инициатору сообщение *Configure-Nak*, в котором содержится список неподтвержденных параметров. Узел-инициатор обрабатывает ответ и отправляет пакет *Configure-Request*, содержащий новые конфигурационные параметры.

Если отвечающий узел полностью отвергает предложенные параметры, он отправляет сообщение *Configure-Reject*. Так же как и в случае получения пакета *Configure-Nak*, узел-инициатор повторяет попытку договориться о параметрах канала и отправляет пакет *Configure-Request*, содержащий новые конфигурационные параметры.

```
⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x01
    Length: 19
    ⊖ Options: (15 bytes)
        Maximum Receive Unit: 1460
    ⊕ Authentication protocol: 5 bytes
        Magic number: 0x77e89c35
```

Рис. 2.5. Пакет Configure-Request

```
⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
    Code: Configuration Ack (0x02)
    Identifier: 0x01
    Length: 19
    ⊖ Options: (15 bytes)
        Maximum Receive Unit: 1460
    ⊕ Authentication protocol: 5 bytes
        Magic number: 0x77e89c35
```

Рис. 2.6. Пакет Configure-Ack

```
⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
    Code: Configuration Reject (0x04)
    Identifier: 0x00
    Length: 7
    ⊖ Options: (3 bytes)
        ⊕ Callback: 3 bytes
```

Рис. 2.7. Пакет Configure-Reject

В фазе установления канала взаимодействующие узлы будут игнорировать все не-LCP пакеты. Узлы вернутся в фазу установления канала, если в фазе аутентификации или пересылки дейтаграмм сетевого уровня одним из них будет получен пакет *Configure-Request*.

Обслуживание канала

Как только переговоры о параметрах канала успешно завершились, LCP переходит в открытое состояние и передает управление соответствующим протоколам аутентификации и/или NCP. В этой фазе пакеты LCP могут

использоваться для управления, поиска неисправностей в канале или тестирования производительности. Пакеты *Code-Reject* и *Protocol-Reject* используются для обратной связи, когда одна из сторон канала получила пакет LCP с неизвестным кодом или кадр PPP с неподдерживаемым типом протокола.

⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
 Code: Echo Request (0x09)
 Identifier: 0x06
 Length: 8
 Magic number: 0x77e89c35

Рис. 2.8. Пакет Echo-Request

⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
 Code: Echo Reply (0x0a)
 Identifier: 0x06
 Length: 8
 Magic number: 0x40ba1152

Рис. 2.9. Пакет Echo-Reply

Для исследования канала в обоих направлениях используются пакеты *Echo-Request* (рис. 2.8) и *Echo-Reply* (рис. 2.9). Пакет *Discard-Request* используется для исследования канала в направлении от отправителя к получателю.

Завершение работы канала

Для завершения работы канала одна из его сторон отправляет пакет *Terminate-Request* (рис. 2.10). Другая сторона отвечает пакетом *Terminate-Ack* (рис. 2.11).

⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
 Code: Termination Request (0x05)
 Identifier: 0x02
 Length: 16
 Data (12 bytes)

Рис. 2.10. Пакет Terminate-Request

⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
 Code: Termination Ack (0x06)
 Identifier: 0x02
 Length: 4

Рис. 2.11. Пакет Terminate-Ack

2.4. Network Control Protocol (NCP)

После завершения фазы установления канала протоколом LCP и аутентификации (которая необязательна и может не выполняться) должны быть независимо друг от друга сконфигурированы протоколы сетевого уровня, такие как IP, IPX и т. п. Конфигурирование параметров протоколов сетевого уровня выполняется протоколами управления сетью (NCP). Для каждого протокола сетевого уровня, пакеты которого передаются по каналу PPP, используется собственный протокол NCP (табл. 2.1). Каждый из протоколов NCP обладает специфическими свойствами, зависящими от соответствующего протокола сетевого уровня, и поддерживает конфигурационные запросы, характерные только для этого протокола. Каждый из протоколов NCP может переходить в открытое состояние и закрываться в любое время.

Таблица 2.1. Стандарты протокола NCP

Номер RFC	Название RFC	Описание
1332	The PPP Internet Protocol Control Protocol (IPCP)	NCP для протокола Internet Protocol (IP)
1377	The PPP OSI Network Layer Control Protocol (OSINLCP)	NCP для набора протоколов сетевого уровня OSI, таких как CNLP, ES-IS и IS-IS
1378	The PPP AppleTalk Control Protocol (ATCP)	NCP для протокола AppleTalk
1552	The PPP Internetworking Packet Exchange Control Protocol (IPXCP)	NCP для протокола Novell Internetworking Packet Exchange (IPX)
2043	The PPP SNA Control Protocol (SNACP)	NCP для протокола IBM Systems Network Architecture (SNA)
2097	The PPP NetBIOS Frames Control Protocol (NBFCP)	NCP для кадров NetBIOS
2472	IP Version 6 over PPP (IPCPv6)	NCP для протокола IPv6

Подобно LCP, каждый протокол NCP выполняет установление, обслуживание и завершение канала. Для выполнения этих функций обе стороны канала обмениваются пакетами протокола NCP. Процесс установки канала и переговоров о параметрах соответствующего протокола сетевого уровня выполняется с помощью пакетов *Configure-Request*, *Configure-Ack*, *Configure-Nak* и *Configure-Reject*, аналогичных LCP. Конфигурационные параметры, передаваемые в этих пакетах, зависят от используемого протокола NCP. Протокол NCP переходит в открытое состояние после согласования параметров, т. е. после получения инициатором пакета *Configure-Ack*. Пакет *Code-Reject* отправляется в том случае, когда одна из сторон канала получила пакет NCP с неизвестным кодом. Для завершения канала одна из его сторон отправляет пакет *Terminate-Request*, на который другая сторона отвечает пакетом *Terminate-Ack*.

Протокол конфигурирования IPv4 (IPCP)

Рассмотрим в качестве примера протокол NCP для IPv4 (IPCP). Протокол управления IP (Internet Protocol Control Protocol, IPCP) предназначен для конфигурирования протокола IPv4 на обоих концах канала PPP. Обмен IPCP-пакетами аналогичен LCP и начинается в состоянии обмена дейтаграммами сетевого уровня (*NETWORK*). IPCP отличается от LCP следующим:

- Поле *Протокол* кадра PPP. Пакет IPCP инкапсулируется в поле *Информация* кадра PPP, поле *Протокол* которого имеет значение 0x8021.
- Поле *Код* пакета IPCP. Используются только коды с номерами 0–7: *Configure-Request*, *Configure-Ack*, *Configure-Nak*, *Configure-Reject*, *Terminate-Request*, *Terminate-Ack*, *Code-Reject*.

Процесс установки канала и переговоров о параметрах протокола IP выполняется с помощью пакетов *Configure-Request*, *Configure-Ack*, *Configure-Nak* и *Configure-Reject*.

Для протокола IP определены две конфигурационные опции, которые могут быть указаны в пакете IPCP *Configure-Request*:

1. *IP-Compression-Protocol* (протокол сжатия IP). Данный параметр позволяет договориться об использовании конкретного протокола сжатия заголовков протоколов TCP и IP. По умолчанию сжатие не требуется.

2. *IP-Address* (IP-адрес). Данный параметр служит для ведения переговоров об IP-адресе, который будет назначен локальной стороне канала. Он позволяет отправителю запроса *Configure-Request* указать желаемый IP-адрес или запросить противоположную сторону выдать ему IP-адрес (рис. 2.12). Противоположная сторона может предоставить данную информацию, отвергнув этот параметр с помощью *Configure-Nak* и указав правильный IP-адрес (рис. 2.13). Также параметр позволяет договориться об IP-адресах первичного и вторичного DNS-серверов, которые будут использоваться на локальной стороне. Когда отвечающий узел соглашается с параметрами, он отправляет пакет *Configure-Ack*.

После того как протокол IPCP перейдет в открытое состояние, по каналу PPP смогут передаваться любые IP-пакеты. Максимальная длина IP-пакета

```
⊕ Point-to-Point Protocol
  ⊕ PPP IP Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x01
    Length: 22
    ⊕ Options: (18 bytes)
      IP address: 0.0.0.0
      Primary DNS server IP address: 0.0.0.0
      Secondary DNS server IP address: 0.0.0.0
```

Рис. 2.12. Запрос IP-адресов протоколом IPCP
(сообщение *Configure-Request*)

```
⊕ Point-to-Point Protocol
⊖ PPP IP Control Protocol
    Code: Configuration Nak (0x03)
    Identifier: 0x01
    Length: 22
    ⊖ Options: (18 bytes)
        IP address: 10.2.1.50
        Primary DNS server IP address: 8.8.8.8
        Secondary DNS server IP address: 8.8.8.8
```

Рис. 2.13. Назначение IP-адресов локальной стороне канала
(сообщение Configure-Nak)

```
⊕ Point-to-Point Protocol
⊖ PPP IP Control Protocol
    Code: Configuration Ack (0x02)
    Identifier: 0x02
    Length: 22
    ⊖ Options: (18 bytes)
        IP address: 10.2.1.50
        Primary DNS server IP address: 8.8.8.8
        Secondary DNS server IP address: 8.8.8.8
```

Рис. 2.14. Подтверждение назначенных IP-адресов (сообщение Configure-Ack)

соответствует максимальной длине поля *Информация* кадра PPP. IP-дейтаграммы большего размера будут фрагментированы. Если необходимо избежать фрагментирования и последующего сбора пакетов, то следует использовать механизм Path MTU discovery. Фрагментация пакетов IPv4 и работа процесса Path MTU discovery будут рассмотрены далее в этом курсе.

2.5. Протоколы аутентификации PPP

Аутентификация в протоколе PPP не является обязательной и выполняется только по согласованию сторон. Взаимодействующие устройства договариваются об используемом протоколе аутентификации в процессе переговоров LCP. В случае успешной договоренности аутентификация выполняется сразу же после установления канала. Продолжение настройки канала при этом будет возможно только после успешной аутентификации.

Первоначально в стеке протоколов PPP были определены два протокола аутентификации: Password Authentication Protocol (PAP) и Challenge Handshake Authentication Protocol (CHAP). Они описаны в RFC 1334 «PPP Authentication Protocols».

Аутентификация (*Authentication*) — сервис безопасности, который обеспечивает подтверждение того, что информация получена от законного источника и получатель является требуемым.

Идентификация (*Identification*) — сервис, с помощью которого определяются уникальные свойства пользователей, которые позволяют отличать их друг от друга, и способы, с помощью которых пользователи указывают свои идентификации информационной системе. Идентификация тесно связана с аутентификацией.

В настоящее время PPP поддерживает несколько протоколов аутентификации:

- Password Authentication Protocol (PAP);
- Challenge Handshake Authentication Protocol (CHAP);
- Microsoft Challenge Handshake Authentication Protocol version 1.0 (MS-CHAP v1);
- Microsoft Challenge Handshake Authentication Protocol version 2.0 (MS-CHAP v2);
- Extensible Authentication Protocol — Transport Level Security (EAP-TLS);
- Protected Extensible Authentication Protocol (PEAP).

В PPP для описания аутентифицируемых узлов используется следующая терминология:

- *Аутентификатор* (*authenticator*) — конец канала, требующий выполнения аутентификации. Аутентификатор определяет протокол аутентификации, который будет указан в пакетах *Configure-Request*, передаваемых в фазе установления канала;
- *Одноранговый узел* (*peer*) — другая сторона канала «точка-точка», чья подлинность проверяется аутентификатором.

2.5.1. Протокол Password Authentication Protocol (PAP)

Протокол PAP — простой протокол двухстороннего рукопожатия (2-way handshake). Процесс аутентификации состоит всего из двух шагов.

1. Запрос аутентификации. Узел отправляет аутентификатору запрос *Authenticate-Request*, который содержит имя пользователя и пароль (рис. 2.15).

2. Ответ на запрос аутентификации. Аутентификатор проверяет полученные имя пользователя и пароль в локальной базе данных или пересыпает их серверу аутентификации (например, RADIUS или TACACS+). Если они достоверны, отправляет узлу сообщение *Authenticate-Ack* (рис. 2.16). Аутентификация успешно завершается, и начинается фаза конфигурирования протоколов сетевого уровня. Если аутентификация неуспешна, аутентификатор отправляет узлу сообщение *Authenticate-Nak* и соединение завершается.

В процессе аутентификации пароль передается в открытом виде, т. е. он не шифруется. В связи с этим аутентификация PAP не является безопасной, поскольку не обеспечивает защиты от различного рода атак.

2. Протокол PPP

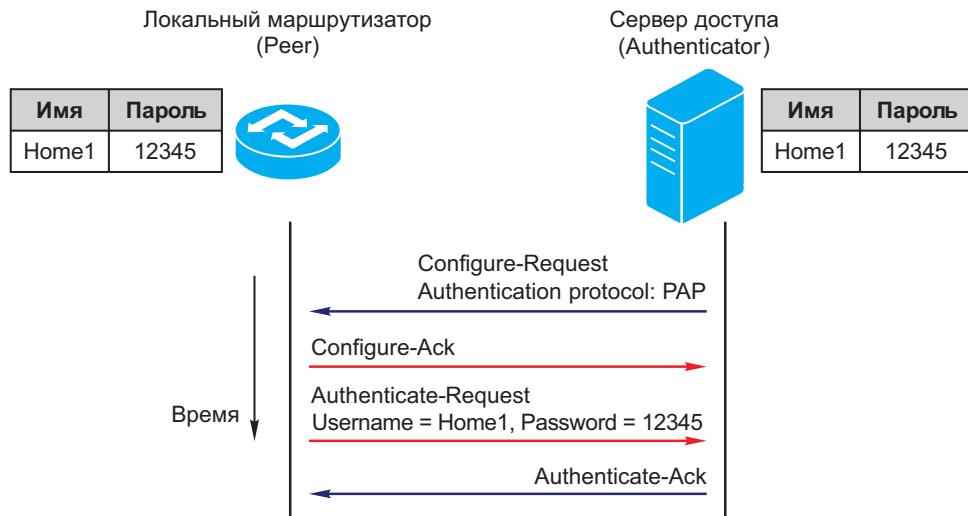


Рис. 2.15. Аутентификация PAP

```
▫ Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Password Authentication Protocol (0xc023)
▫ PPP Password Authentication Protocol
  Code: Authenticate-Request (1)
  Identifier: 2
  Length: 14
▫ Data
  Peer-ID-Length: 4
  Peer-ID: test
  Password-Length: 4
  Password: test
```

Рис. 2.16. Запрос аутентификации PAP (Authenticate-Request)

```
▫ Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Password Authentication Protocol (0xc023)
▫ PPP Password Authentication Protocol
  Code: Authenticate-Ack (2)
  Identifier: 2
  Length: 13
▫ Data
  Message-Length: 8
  Message: Login ok
```

Рис. 2.17. Ответ на запрос аутентификации PAP (Authenticate-Ack)

2.5.2. Протокол Challenge Handshake Authentication Protocol (CHAP)

Протокол CHAP используется для периодической проверки аутентификации противоположной стороны с помощью трехстороннего рукопожатия (3-way handshake). Протокол выполняется после установления канала и может быть повторно использован в любое время после того, как канал установлен. Первоначально CHAP был описан в RFC 1334, а позднее дополнен в RFC 1994.

Алгоритм DES (*Data Encryption Standard*) — алгоритм симметричного шифрования, длина блока равна 64 битам, длина ключа равна 56 битам.

Атаки повторного использования (*replay-атаки*) — пассивный захват данных с последующей их пересылкой целевой системе для получения несанкционированного доступа.

Хеш-функция — функция, отображающая данные произвольной длины в строку битов фиксированной длины, которая может использоваться для аутентификации исходных данных. Хеш-код — результат, создаваемый хеш-функцией.

Процесс аутентификации CHAP показан на рис. 2.18.

Аутентификация состоит из следующих шагов:

- После завершения фазы установления канала аутентификатор посыпает узлу сообщение *Challenge* (рис. 2.19). Оно содержит идентификатор (Identifier)

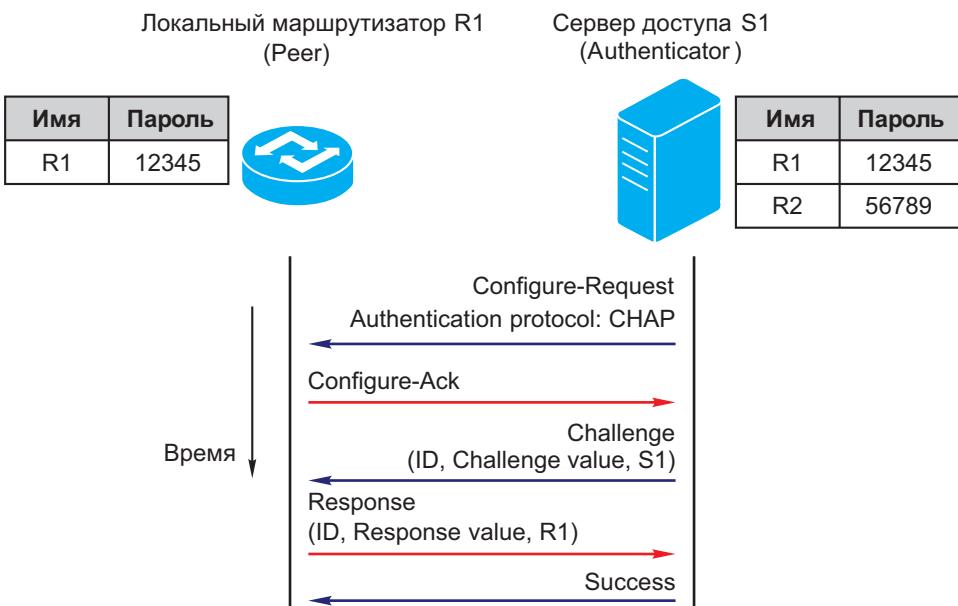


Рис. 2.18. Аутентификация CHAP

2. Протокол PPP

```
« Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Challenge Handshake Authentication Protocol (0xc223)
« PPP Challenge Handshake Authentication Protocol
  Code: Challenge (1)
  Identifier: 195
  Length: 32
« Data
  Value Size: 22
  Value: 4382e49079bcb8595d6358e1fc926857e69386c08431
  Name: l2tpd
```

Рис. 2.19. Сообщение Challenge CHAP

этого сообщения и вызов (Challenge Value) — произвольную строку символов переменной длины.

- Узел-получатель выполняет одностороннюю хеш-функцию (обычно Message Digest 5 (MD5)) над информацией, полученной из сообщения *Challenge* (идентификатором и вызовом) и паролем. Вычисленное значение он отправляет аутентификатору в сообщении *Response* (рис. 2.20). Идентификатор сообщения *Response* аналогичен идентификатору сообщения *Challenge*.

```
« Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Challenge Handshake Authentication Protocol (0xc223)
« PPP Challenge Handshake Authentication Protocol
  Code: Response (2)
  Identifier: 195
  Length: 25
« Data
  Value Size: 16
  Value: 9d3af2f89916945e96fa315ba4a9918a
  Name: test
```

Рис. 2.20. Сообщение Response CHAP

- Аутентификатор сравнивает полученное значение с собственным вычисленным значением хеш-функции. Хеш-функция на стороне аутентификатора выполняется над идентификатором и вызовом оригинального сообщения *Challenge* и паролем. Поэтому у аутентификатора и узла должны быть одинаковые пароли. Если вычисленное и полученное значения хеш-функции совпали, аутентификация считается успешной и аутентификатор отправляет сообщение *Success* (рис. 2.21). В противном случае соединение сбрасывается и отправляется сообщение *Failure*.

```
▲ Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Challenge Handshake Authentication Protocol (0xc223)
▲ PPP Challenge Handshake Authentication Protocol
  Code: Success (3)
  Identifier: 195
  Length: 18
  Message: Access granted
```

Рис. 2.21. Сообщение Success CHAP

- Через произвольные интервалы времени аутентификатор отправляет новый *Challenge* и шаги 1–3 повторяются.

CHAP обеспечивает защиту от replay-атак благодаря использованию при каждой отправке сообщения *Challenge* уникального значения идентификатора и случайного набора символов переменной длины. Таким образом, нет возможности воспользоваться перехваченной информацией для получения неавторизованного доступа к сети. Повторная отправка сообщений *Challenge* в течение сессии CHAP предназначена для ограничения времени на проведение любых единичных атак. Аутентификатор управляет частотой и временем отправки сообщений *Challenge*.

Метод аутентификации CHAP основан на том, что оба участника знают некий секрет (пароль), который никогда не отправляется по каналу. Хотя аутентификация односторонняя, стороны могут договориться о применении CHAP в обоих направлениях и использовать одинаковый пароль для обеспечения взаимной аутентификации. Так как CHAP может использоваться для аутентификации множества различных устройств, поле *Имя* (Name) в сообщениях *Challenge* и *Response* может служить индексом для поиска правильно-го пароля в больших таблицах, хранящих их. Также это позволяет создавать для каждого устройства более одной пары имя/пароль и изменять используемый пароль в любое время в течение сессии.

Аутентификация CHAP имеет свои недостатки, в частности требует, чтобы пароль хранился в открытом виде. При этом она устраниет одну из уязвимостей PAP — пароль не передается по сети в открытом виде, а используются вызов и хеширование ответа, что позволяет избежать replay-атак. Также, в отличие от PAP, аутентификация CHAP может выполняться несколько раз за сессию. В 1996 году IETF обновила стандарт, описывающий аутентификацию PPP, и оставила в нем только CHAP. Новый стандарт получил название RFC 1994 «PPP Challenge Handshake Authentication Protocol (CHAP)». Стандарт RFC 1334 в настоящее время является устаревшим.

2.5.3. Протокол Microsoft Challenge Handshake Authentication Protocol (MS-CHAP)

MS-CHAP (Microsoft Challenge Handshake Authentication Protocol) был создан компанией Microsoft для аутентификации удаленных рабочих станций Windows. Существуют две версии этого протокола: MS-CHAP v1 и MS-CHAP v2.

Первая версия протокола MS-CHAP описана в RFC 2433. Там, где это возможно, она согласуется со стандартом CHAP. Различия между MS-CHAP v1 и стандартным CHAP следующие:

- Согласование MS-CHAP в процессе переговоров LCP аналогично согласованию стандартного CHAP, за исключением того, что поле *Algorithm* опции *Authentication Protocol* имеет значение 0x80.
- Пакет MS-CHAP Response представлен в формате, обеспечивающем совместимость с Microsoft Windows NT 3.5, 3.51 и 4.0 и Windows 95. Формат MS-CHAP не требует хранения аутентификатором пароля в открытом или в обратимо зашифрованном виде.
- MS-CHAP обеспечивает механизмы запроса аутентифицирующей стороной повторной аутентификации и изменения пароля.
- MS-CHAP определяет набор значений кодов, сообщающих о причине отказа, которые возвращаются узлу в сообщении *Failure* при неудачной аутентификации.

Аутентификация MS-CHAP v1 состоит из трех шагов:

1. Аутентификатор отправляет сообщение MS-CHAP *Challenge*, содержащее 8-байтную произвольную строку символов (вызов).
2. Узел отправляет сообщение MS-CHAP *Response*. По сравнению с CHAP его поле *Value* отформатировано по-другому:

- 24 октета: ответ на запрос, совместимый с LAN Manager
- 24 октета: ответ на запрос, совместимый с Windows NT
- 1 октет: флаг совместимости с Windows NT

Ответ на запрос, совместимый с LAN Manager, является результатом функции шифрования `LmChallengeResponse()`, выполненной над хешем пароля в формате LAN Manager и полученной 8-байтной строки символов. Узел использует хеш пароля в формате LAN Manager, чтобы получить три ключа алгоритма DES. Каждый из этих ключей применяется для шифрования полученного вызова. Получаются три зашифрованных блока, которые объединяются в 24-байтный ответ.

Ответ на запрос, совместимый с Windows NT, является результатом функции шифрования `NTChallengeResponse()`, выполненной над хешем пароля в формате Windows NT и полученной 8-байтной строки символов. Узел создает второй 24-байтный ответ, используя хеш пароля в формате Windows NT и аналогичную процедуру.

Если флаг совместимости с Windows NT равен 1, это указывает на то, что поддерживается ответ в формате Windows NT и он предпочтительнее ответа, совместимого с LAN Manager.

3. Аутентификатор расшифровывает полученный ответ, используя известный ему хеш пароля, совместимый с LAN Manager или с Windows NT, в зависимости от установленного в ответе флага. Если расшифрованные блоки совпадают с отправленным вызовом, аутентификация считается успешной и аутентификатор отправляет сообщение *Success*. В противном случае отправляется сообщение *Failure*.

По сравнению с CHAP в MS-CHAP v1 появилось новое сообщение *Change Password*, которое позволяет узлу изменить пароль, указанный в последнем отправленном пакете *Response*.

Основным недостатком MS-CHAP v1 является отправка клиентом ответа, содержащего две величины: в формате совместимости с LAN Manager и в формате совместимости с Windows NT.

Несмотря на то что по сравнению с CHAP в MS-CHAP v1 появилось шифрование ответа, он не является безопасным протоколом аутентификации. Для формирования ответа вызов шифруется с помощью ключей DES, основанных на хеше пароля. Возможность взлома пароля зависит от криптографической стойкости используемой хеш-функции. Хеш в формате LAN Manager значительно слабее хеша формата Windows NT. Всякий раз, когда пользователь подключается с одним и тем же паролем, создаются одни и те же ключи шифрования. Поэтому в средах с большой вероятностью прослушивания необходимо достаточно часто менять пароль.

Протокол MS-CHAP версии 2 (описан в RFC 2759) соответствует как протоколу MS-CHAP v1, так и стандартному протоколу CHAP. Различия между MS-CHAP v1 и MS-CHAP v2 (рис. 2.22) следующие:

- Согласование MS-CHAP v2 в процессе переговоров LCP аналогично согласованию стандартного CHAP, за исключением того, что поле *Algorithm* опции *Authentication Protocol* имеет значение 0x81.
- Протокол MS-CHAP v2 обеспечивает взаимную аутентификацию участников, добавляя в сообщении *Response* вызов узла (*Peer-Challenge*), на который аутентификатор отвечает в сообщении *Success*.
- В MS-CHAP v2 больше не поддерживается формат LAN Manager. Ответ в этом формате заменен в сообщении *Response* на *Peer-Challenge*.
- Изменен формат поля *Message* в пакете *Failure*.

```
⊕ Point-to-Point Protocol
⊖ PPP Link Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x01
    Length: 19
    ⊖ Options: (15 bytes)
        Maximum Receive Unit: 1460
    ⊖ Authentication protocol: 5 bytes
        Authentication protocol: Challenge Handshake Authentication Protocol (0xc223)
        Algorithm: MS-CHAP-2 (0x81)
        Magic number: 0x77e89c35
```

Рис. 2.22. Параметр настройки MS-CHAP v2 в пакете LCP

2. Протокол PPP

- Пакеты *Change Password* версий 1 и 2 больше не поддерживаются. Они заменены единым пакетом *Change-Password*.

Аутентификация MS-CHAP v2 состоит из следующих шагов:

1. Аутентификатор отправляет узлу сообщение MS-CHAP v2 *Challenge* (рис. 2.23), которое содержит идентификатор и 16-байтную произвольную строку символов (вызов).



Рис. 2.23. Сообщение MS-CHAP v2 Challenge

2. Узел посыпает ответ MS-CHAP v2 *Response* (рис. 2.24). По сравнению с CHAP его поле *Value* отформатировано по-другому:

- 16 октетов: произвольная строка символов, сгенерированных узлом (*Peer-Challenge*);
 - 8 октетов: зарезервировано;
 - 24 октета: ответ на запрос *NT-Response*;
 - 1 октет: флаги.

Ответ на запрос *NT-Response* является результатом функции шифрования *GenerateNTResponse()*, входными данными для которой являются пароль, имя пользователя, строка *Peer-Challenge* и полученная из сообщения *Challenge* 16-байтная строка символов. Шифрование выполняется с помощью алгоритма DES.

Аутентификатор отправляет узлу 16-байтный вызов, но, в отличие от MS-CHAP v1, узел его напрямую не использует. Он получает из него 8-байтный вызов следующим образом:

а. Узел генерирует произвольную строку символов, называемую *Peer-Challenge*.

б. Стока *Peer-Challenge* объединяется с 16-байтным вызовом, полученным от аутентификатора и именем пользователя.

в. Полученный результат хешируется с помощью SHA-1.

г. Первые 8 байт полученного хеша становятся 8-байтным вызовом.

Далее узел создает 24-байтный ответ, используя хеш-функцию Windows NT и полученный 8-байтный вызов. Процедура получения ответа аналогична MS-CHAP v1.

3. Аутентификатор расшифровывает полученный *NT-Response*, используя известный ему хеш пароля узла. Если расшифрованные блоки совпадают с отправленным вызовом, аутентификация считается успешной,

```
⊕ Point-to-Point Protocol
⊖ PPP Challenge Handshake Authentication Protocol
    Code: Response (2)
    Identifier: 174
    Length: 61
    ⊖ Data
        Value Size: 49
        Value: 1bb0d79c1d96c1c1413530d9aa189f04335049bc7f736964...
        Name: Client1
```

Рис. 2.24. Сообщение MS-CHAP v2 Response

и аутентификатор отправляет сообщение *Success* (рис. 2.25). В противном случае отправляется сообщение *Failure*. Сообщение *Success* в поле *Message* содержит ответ аутентификатора длиной 20 байт, представленный в кодировке ASCII в виде 40 шестнадцатеричных чисел и текстовую строку.

Ответ аутентификатора получается следующим образом:

а. Аутентификатор хранит хешированный с помощью MD4 пароль узла. Он хеширует этот хеш пароля с использованием MD4, чтобы получить хеш хеша пароля.

б. Аутентификатор объединяет хеш хеша пароля, 24-байтный ответ *NT-Response* и константу «*Magic1*» и хеширует полученный результат с помощью SHA.

с. К полученному хешу присоединяются 8-байтный вызов из ответа на запрос и константа «*Magic2*». Результат хешируется с помощью SHA.

Полученный ответ используется для взаимной аутентификации узла и аутентификатора.

```
⊕ Point-to-Point Protocol
⊖ PPP Challenge Handshake Authentication Protocol
    Code: Success (3)
    Identifier: 174
    Length: 63
    Message: S=4749880BD55F2FE30D78C26FFA43AABC8D7E431A M=Access granted
```

Рис. 2.25. Сообщение MS-CHAP v2 Success

4. Узел проверяет ответ аутентификатора, проводя аналогичные вычисления и сравнивая полученный ответ с вычисленным. Если они не совпали, узел прерывает соединение.

2.6. Сжатие данных в PPP

Протокол PPP включает дополнительную функцию сжатия данных, которая позволяет повысить производительность на медленных последовательных линиях связи. Об использовании этой функции стороны могут

договориться в процессе переговоров LCP. За настройку, активизацию и отключение алгоритмов сжатия на обоих концах канала PPP отвечает протокол *Compression Control Protocol* (CCP) (рис. 2.26). Он использует механизм обмена пакетами аналогичный LCP. Протокол CCP, так же как и NCP, начинает обмен пакетами в фазе пересылки дейтаграмм сетевого уровня (*NETWORK*). Процесс настройки сжатия и переговоров о параметрах выполняется с помощью пакетов *Configure-Request*, *Configure-Ack*, *Configure-Nak*, *Configure-Reject*. Конфигурационные параметры, передаваемые в этих пакетах, специфичны для протокола CCP. Пакет *Code-Reject* отправляется в том случае, когда одна из сторон канала получила пакет CCP с неизвестным кодом. Два новых типа сообщений *Reset-Request* и *Reset-Ack* используются для прекращения сжатия в случае обнаружения проблем с восстановлением сжатых данных одной из сторон. Для завершения CCP одна из его сторон отправляет пакет *Terminate-Request*, на который другая сторона отвечает пакетом *Terminate-Ack*. Закрытие CCP не приведет к закрытию канала PPP. После того как CCP перейдет в открытое состояние, PPP начинает передачу сжатых пакетов.

```
+ Point-to-Point Protocol
  - PPP Compression Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x01
    Length: 4
```

Рис. 2.26. Запрос протокола CCP

Конфигурационные опции CCP позволяют договориться об алгоритмах сжатия и их параметрах. В настоящее время поддерживаются следующие алгоритмы сжатия:

- PPP Predictor Compression Protocol (RFC 1978)
- Puddle Jumper (RFC 1962)
- Hewlett-Packard PPC (RFC 1962)
- PPP Stac LZS Compression Protocol (RFC 1974)
- Microsoft Point-To-Point Compression (MPPC) Protocol (RFC 2118)
- PPP Gandalf FZA Compression Protocol (RFC 1993)
- V.42bis compression (RFC 1962)
- PPP BSD Compression Protocol (RFC 1977)
- PPP LZS-DCP Compression Protocol (LZS-DCP) (RFC 1967)
- PPP Deflate Protocol (RFC 1979)

2.7. Протоколы шифрования данных PPP

Протоколы аутентификации гарантируют, что только авторизованные устройства могут установить соединение PPP. После установления соединения данные передаются по каналу в открытом виде. Шифрование данных, как

и аутентификация, является опциональной функцией. Стороны могут договориться об используемом алгоритме шифрования с помощью протокола *Encryption Control Protocol* (ECP).

Протокол ECP отвечает за конфигурацию и активизацию алгоритмов шифрования на обоих концах канала PPP. Он использует механизм обмена пакетами, аналогичный LCP, который начинается в фазе пересылки дейтаграмм сетевого уровня (*NETWORK*). Если используется аутентификация или выполняется проверка качества канала, протокол ECP должен ожидать завершения этих процессов.

Аналогично LCP, протокол ECP выполняет установление, обслуживание и завершение канала. Для выполнения этих функций обе стороны канала обмениваются пакетами протокола ECP. Процесс установки канала и переговоров об алгоритмах шифрования и их параметрах выполняется с помощью пакетов *Configure-Request*, *Configure-Ack*, *Configure-Nak* и *Configure-Reject*, аналогичных LCP. Конфигурационные параметры, передаваемые в этих пакетах, специфичны для протокола ECP. Пакет *Code-Reject* отправляется в том случае, когда одна из сторон канала получила пакет ECP с неизвестным кодом. Два новых типа сообщений *Reset-Request* и *Reset-Ack* используются для прекращения шифрования в случае обнаружения проблем с дешифрованием одной из сторон. Для завершения ECP одна из его сторон отправляет пакет *Terminate-Request*, на который другая сторона отвечает пакетом *Terminate-Ack*. Закрытие ECP не приведет к закрытию канала PPP.

После того как ECP перейдет в открытое состояние, PPP начинает передачу шифрованных пакетов.

Конфигурационные опции ECP используются для определения типа алгоритма шифрования, который будет использоваться сторонами. В настоящее время ECP поддерживает два алгоритма шифрования — 3DESE (RFF 2420 «The PPP Triple-DES Encryption Protocol (3DESE)») и DESE-bis (RFF 2419 «The PPP DES Encryption Protocol, Version 2 (DESE-bis)»). Также предусмотрена возможность использования собственных протоколов производителей.

2.7.1. Протокол шифрования MPPE

Протокол *Microsoft Point-to-Point Encryption* (MPPE) разработан компанией Microsoft для шифрования пакетов PPP (RFC 3078). MPPE работает как подфункция протокола Microsoft Point-To-Point Compression (MPPC). Этот протокол служит для сжатия данных при их передаче по каналу PPP. Протокол MPPC является опцией протокола Compression Control Protocol, поэтому переговоры о параметрах MPPE происходят внутри опции 18 протокола CCP (рис. 2.27).

Для обеспечения конфиденциальности данных в протоколе используется алгоритм шифрования RC4. В настоящее время MPPE поддерживает сессионные ключи длиной 40, 56 и 128 бит. Стороны могут договориться

2. Протокол PPP

о длине ключа, используемой для шифрования, в процессе переговоров (рис. 2.28). Инициатор переговоров отправляет сообщение *Configure-Request* и указывает в опциях все алгоритмы, которые он поддерживает.

```
Ethernet II, Src: D-LinkIn_f0:cc:ef (90:94:e4:f0:cc:ef), Dst: WistronI_73:7c:8c (20:6a:8a:73:7c:8c)
Internet Protocol Version 4, Src: 172.16.5.1, Dst: 172.16.5.3
User Datagram Protocol, Src Port: 1701, Dst Port: 1701
Layer 2 Tunneling Protocol
Point-to-Point Protocol
    Address: 0xff
    Control: 0x03
    Protocol: Compression Control Protocol (0x80fd)
PPP Compression Control Protocol
    Code: Configuration Request (1)
    Identifier: 1 (0x01)
    Length: 10
    ▾ Options: (6 bytes), Microsoft PPE/PPC
        ▾ Microsoft PPE/PPC
            Type: Microsoft PPE/PPC (18)
            Length: 6
            ▾ Supported Bits: 0x01000020, H, L
                .... .... 1 .... .... .... .... = H: Stateless mode ON
                .... .... .... .... 0.... .... = M: 56-bit encryption OFF
                .... .... .... .... .0.... .... = S: 128-bit encryption OFF
                .... .... .... .... ..1.... .... = L: 40-bit encryption ON
                .... .... .... .... ...0 .... = D: Obsolete (should ALWAYS be 0)
                .... .... .... .... ...0 .... = C: No desire to negotiate MPPE
```

Рис. 2.27. Переговоры об используемой длине ключа

Получатель отвечает сообщением *Configure-Nak* с единственной установленной опцией. Если получатель поддерживает несколько алгоритмов шифрования, то указывается наиболее сильный. Затем инициатор должен либо отправить другой запрос *Configure-Request*, содержащий ту же самую опцию, что и в полученном сообщении *Configure-Nak*, либо прервать переговоры и сбросить соединение. В случае согласия с конфигурацией получатель отправляет сообщение *Configure-Ack* (рис. 2.28).

MPPE требует использования ключей шифрования, генерируемых в процессе аутентификации по протоколу MS-CHAP v1, MS-CHAP v2 или EAP-TLS (Extensible Authentication Protocol-Transport Level Security). Чтобы включить шифрование данных с использованием MPPE, на устройствах необходимо активизировать проверку подлинности по протоколу MS-CHAP v1, MS-CHAP v2 или EAP-TLS (рис. 2.29). Все эти методы проверки подлинности генерируют ключи на основе пароля пользователя.

Для того чтобы передавать зашифрованные с помощью MPPE пакеты, протокол PPP должен перейти в состояние обмена дейтаграммами сетевого уровня. В поле *Информация* кадра PPP инкапсулируется одна MPPE-дейтаграмма. Для всех зашифрованных дейтаграмм в поле *Протокол* указан тип 0x00FD (рис. 2.30).

```
Ethernet II, Src: D-LinkIn_f0:cc:ef (90:94:e4:f0:cc:ef), Dst: WistronI_73:7c:8c (20:6a:8a:73:7c:8c)
Internet Protocol Version 4, Src: 172.16.5.1, Dst: 172.16.5.3
User Datagram Protocol, Src Port: 1701, Dst Port: 1701
Layer 2 Tunneling Protocol
Point-to-Point Protocol
    Address: 0xff
    Control: 0x03
    Protocol: Compression Control Protocol (0x80fd)
PPP Compression Control Protocol
    Code: Configuration Nak (3)
    Identifier: 7 (0x07)
    Length: 10
    Options: (6 bytes), Microsoft PPE/PPC
        Microsoft PPE/PPC
            Type: Microsoft PPE/PPC (18)
            Length: 6
        Supported Bits: 0x01000021, H, L, C
            .... ...1 ..... .... .... .... .... = H: Stateless mode ON
            .... .... .... .... 0.... .... .... = M: 56-bit encryption OFF
            .... .... .... .... .0.... .... .... = S: 128-bit encryption OFF
            .... .... .... .... ..1.... .... .... = L: 40-bit encryption ON
            .... .... .... .... ...0 .... .... .... = D: Obsolete (should ALWAYS be 0)
            .... .... .... .... ....1 .... .... .... = C: Desire to negotiate MPPC
```

Рис. 2.28. Завершение переговоров об используемой длине ключа

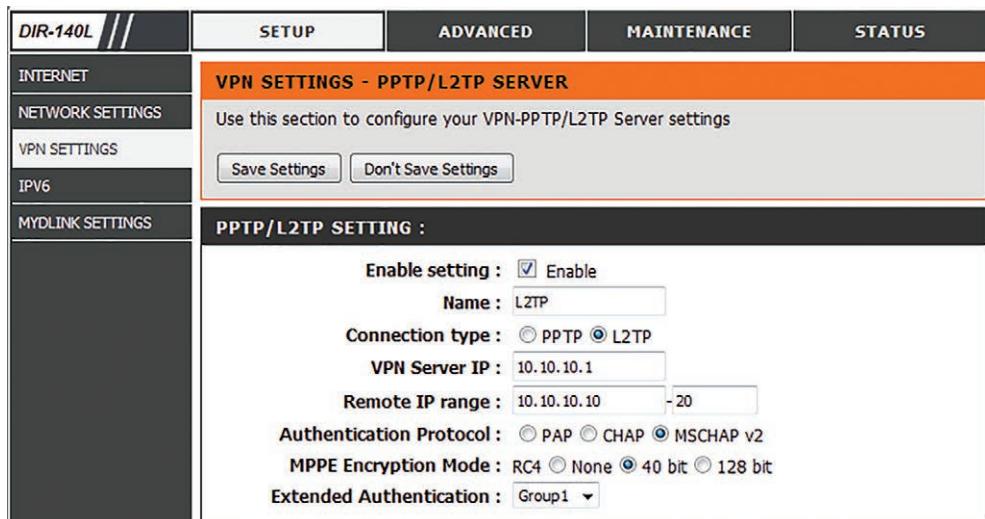


Рис. 2.29. Пример настройки шифрования MPPE

2. Протокол PPP

```
Ethernet II, Src: WistronI_73:7c:8c (20:6a:8a:73:7c:8c), Dst: D-LinkIn_f0:cc:ef (90:94:e4:f0:cc:ef)
Internet Protocol Version 4, Src: 172.16.5.3, Dst: 172.16.5.1
User Datagram Protocol, Src Port: 1701, Dst Port: 1701
Layer 2 Tunneling Protocol
Point-to-Point Protocol
  Address: 0xff
  Control: 0x03
  Protocol: Compressed datagram (0x00fd)
PPP Compressed Datagram
```

Рис. 2.30. Передача зашифрованных дейтаграмм в протоколе PPP

Внимание: протокол MPPE имеет известные принципиальные уязвимости, позволяющие взломать его, поэтому он обеспечивает весьма слабую защиту и его не рекомендуется использовать для шифрования критически важных данных.

2.8. Протоколы туннелирования PPP

Виртуальные частные сети (*Virtual Private Network, VPN*) широко используются для организации подключения пользователей к сетям провайдеров услуг, доступа удаленных пользователей в локальную сеть или соединения удаленных локальных сетей одной организации. VPN — это набор технологий, которые позволяют создавать логические сети, использующие сети других сетевых протоколов в качестве транспорта. VPN-соединения могут создаваться на различных уровнях модели OSI. Создаваемая логическая сеть не обязательно должна быть маршрутизируемой, она может обеспечивать соединение типа «точка-точка». Характеристики безопасности созданной логической сети могут отличаться от характеристик безопасности транспортной сети. При создании VPN всегда следует помнить, что безопасность компьютерной системы и сетевого трафика зависит от многих факторов. Разворачивание VPN с использованием той или иной технологии является только частью комплексного подхода к обеспечению безопасности. Безопасность, обеспечиваемая VPN, зависит от многих параметров операционного окружения, в котором VPN выполняется. Например, от безопасности ОС, источника случайных чисел, способов управления системой и т. д.

Виртуальные частные сети второго уровня модели OSI представлены соединениями PPP over Ethernet (PPPoE), PPP over AAL5 (PPPoA), Point-to-Point Tunneling Protocol (PPTP) и Layer Two Tunneling Protocol (L2TP). Эти протоколы представляют собой расширения PPP, позволяющие устанавливать соединения «точка-точка» через сети Ethernet, ATM и IP. Все эти протоколы имеют ряд свойств, унаследованных от PPP:

- Соединения имеют сеансовый, а не статический характер.
- При установке PPP-соединения могут быть выполнены аутентификация и авторизация (односторонняя или взаимная), а по мере работы — учет

работы пользователей. Для этих целей может быть использована локальная база данных пользователей или централизованные серверы TACACS+ и RADIUS.

- При установке IP-соединения могут быть согласованы параметры IP.
- При передаче трафика могут использоваться сжатие с помощью различных методов, а также защита по протоколу MPPE.

Перечисленные свойства полностью обеспечивают решение задач, определяющих сущность VPN: аутентификацию сторон, сохранение целостности данных и защиту от несанкционированного доступа к ним.

Протоколы PPPoE, PPTP и L2TP широко применяются в сетях доступа для аутентификации, авторизации и учета работы клиентов городских и домовых сетей Ethernet и xDSL. PPTP и L2TP также удобны для решения задач удаленного доступа сотрудников к корпоративной сети и соединения локальных сетей головного офиса и филиалов через сети общего пользования (как проводные, так и беспроводные). После подключения к Интернет клиентское устройство доступа инициирует со своей стороны установление туннеля до центрального маршрутизатора корпоративной сети. Через этот туннель узлы, расположенные в головном офисе, уже могут инициировать взаимодействие с филиалом.

Существенное отличие PPPoE, PPPoA, PPTP и L2TP от PPP заключается в том, что эти протоколы — асимметричные, т. е. они основаны на четком разделении ролей клиента и сервера. Клиент всегда инициирует соединение со своей стороны, а сервер находится в режиме пассивного ожидания соединений. Как правило, сервер динамически назначает клиенту параметры IP.

2.9. Передача PPP через Ethernet

Современные технологии доступа предназначены для нескольких конфликтующих друг с другом целей. С одной стороны, желательно установить соединение с несколькими узлами через одно и то же устройство доступа. С другой стороны, необходимо обеспечить управление доступом и возможности биллинга, аналогичные имеющимся у Dial-Up сервисов, использующих PPP. Во многих технологиях доступа подключение нескольких узлов к устройству доступа выполняется через Ethernet.

Протокол Point-to-Point Protocol over Ethernet (PPPoE) обеспечивает способ передачи кадров PPP через Ethernet и позволяет подключать сетевые узлы локальной сети через одно абонентское устройство к концентратору доступа (Access Concentrator, AC), расположенному на стороне провайдера.

Абонентское устройство (*Customer Premises Equipment, CPE*) — пользовательское (оконечное) оборудование, подключаемое к сети связи.

Основную идею работы PPPoE можно объяснить следующим образом. Он моделирует телефонное соединение, в котором каждая сессия PPP трактуется как отдельная телефонная линия. Чтобы обеспечить соединение

2. Протокол PPP

«точка-точка» через Ethernet, каждая сессия PPP должна знать MAC-адрес удаленного узла и установить уникальный идентификатор сессии. Эти задачи в PPPoE выполняет протокол обнаружения. В то же время PPPoE не позволяет пользователю постоянно оставаться на связи. Он должен сообщать свое имя и пароль каждый раз, когда хочет подключиться к сети.

Протокол PPPoE использует клиент-серверную модель. Сервером является концентратор доступа. Роль клиента может выполнять узел или абонентское устройство, которым обычно является маршрутизатор. В связи с этим существуют две сетевые модели PPPoE.

1. Сессия PPP устанавливается между узлом, подключенным к абонентскому устройству, и сервером PPPoE (рис. 2.32). На узле должно быть установлено программное обеспечение клиента PPPoE. В настройках клиента PPPoE указываются имя пользователя и пароль, предоставленные провайдером. Маршрутизатор работает в качестве прозрачного моста (RFC 1483 bridging), т. е. передает кадры от узла в сеть провайдера. Для этого в его настройках надо активировать функцию PPPoE pass through (Проброс PPPoE). Эта функция разрешает маршрутизатору пропускать PPPoE-трафик из локальной сети и в нее.

Если в сети несколько узлов, и каждый из них надо подключить к Интернету, то провайдер должен назначить каждому узлу индивидуальные имя пользователя и пароль. Сессия PPP будет устанавливаться между каждым узлом и сервером доступа. Данная сетевая модель обычно используется для организации доступа в Интернет одного узла, так как подключение множества узлов требует от администратора значительных временных затрат на настройку, мониторинг и контроль каждого из них, а также увеличивает стоимость подключения.

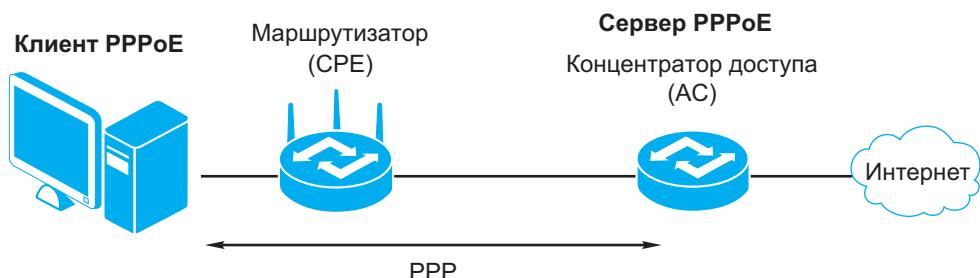


Рис. 2.31. PPPoE-соединение между узлом и концентратором доступа

2. Сессия PPP устанавливается между абонентским устройством и концентратором доступа провайдера. В этой модели роль клиента PPPoE исполняет абонентское устройство. Другими словами, PPPoE-клиент реализован в маршрутизаторе. Такой режим его работы называется «режимом маршрутизации» (RFC 1483 routing). Все узлы локальной сети, подключающиеся

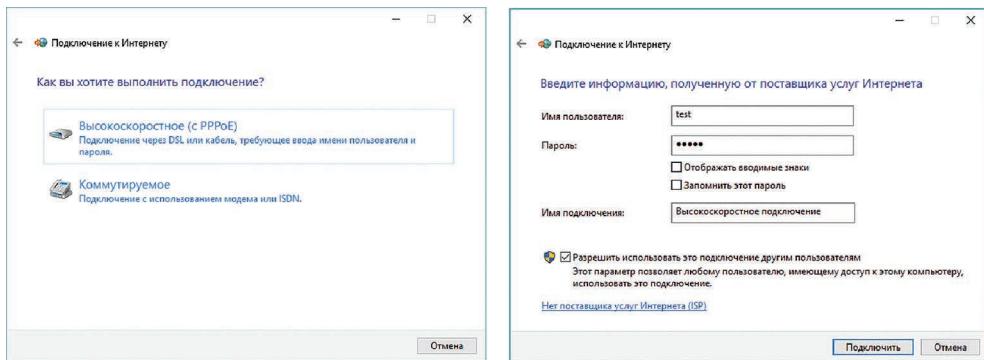


Рис. 2.32. Настройка PPPoE-соединения в Windows

к маршрутизатору по Ethernet или Wi-Fi, передают данные, используя одно PPPoE-соединение, которое установлено между ним и концентратором доступа (рис. 2.33). В настройках PPPoE-соединения на маршрутизаторе указываются имя пользователя и пароль, предоставленные провайдером. При этом не требуется, чтобы на узлах было установлено программное обеспечение PPPoE-клиента (рис. 2.34). Решение о перенаправлении пакетов маршрутизатор принимает на основе информации, хранимой в его таблице маршрутизации. Данная сетевая модель используется при подключении к Интернету множества устройств с использованием одной учетной записи.

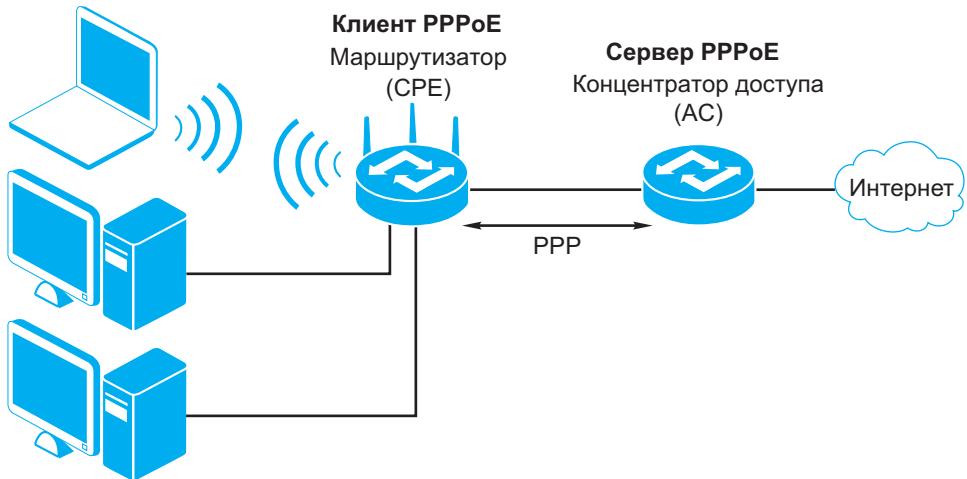


Рис. 2.33. PPPoE-соединение между маршрутизатором и концентратором доступа (режим маршрутизации)

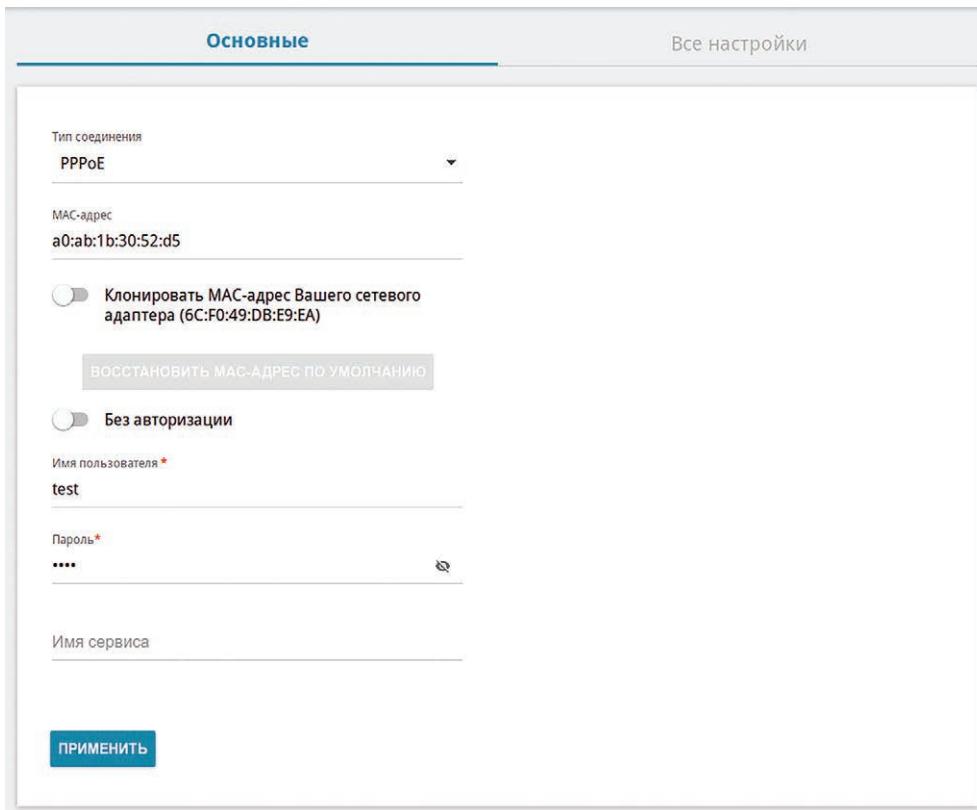


Рис. 2.34. Пример настройки PPPoE-соединения на маршрутизаторе D-Link

Метод доступа PPPoE получил широкое распространение у провайдеров. Так как каждый клиент идентифицируется собственным именем и паролем, провайдеры могут выполнять управление доступом, биллинг, предоставление различных типов сервиса для каждого пользователя в отдельности.

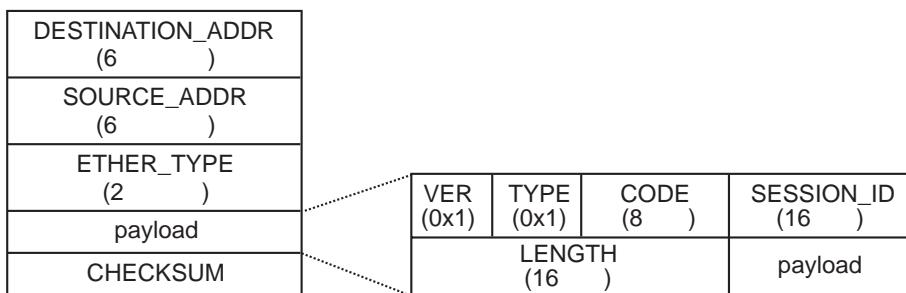
Формат пакета PPPoE

Пакет PPPoE (рис. 2.35) представляет собой кадр PPP, инкапсулированный в кадр Ethernet.

Ниже перечислены поля пакета.

1. Поле DESTINATION_ADDR содержит уникальный адрес Ethernet получателя или широковещательный адрес Ethernet (0xffffffff). Для пакетов стадии Discovery это поле является или уникальным, или широковещательным адресом. Клиент PPPoE использует широковещательный адрес для обнаружения сервера PPPoE. После обнаружения сервера для передачи трафика сессии PPP используется уникальный адрес, определенный на стадии Discovery.

2. Поле SOURCE_ADDR содержит MAC-адрес устройства-отправителя.

**Рис. 2.35.** Формат пакета PPPoE

3. Поле ETHER_TYPE имеет значения 0x8863 (стадия Discovery) или 0x8864 (стадия PPP Session).

В поле данных кадра Ethernet (payload) помещен пакет PPPoE. Он имеет следующие поля:

4. Поле VER длиной 4 бита определяет версию протокола PPPoE и должно иметь значение 0x1.

5. Поле TYPE длиной 4 бита имеет значение 0x1 для данной версии спецификации PPPoE.

6. Поле CODE длиной 8 битов определяет типы пакетов, отправляемых в процессе выполнения стадий Discovery и PPP Session.

7. Поле SESSION_ID имеет длину 16 битов. Оно определяет уникальный идентификатор сессии PPP. Значение поля фиксировано для каждой сессии и фактически определяет PPP-сессию наряду с полями SOURCE_ADDR и DESTINATION_ADDR. Значение 0xffff зарезервировано на будущее и не используется.

8. Поле LENGTH имеет длину 16 битов и определяет длину поля данных (payload) пакета PPPoE. Оно не включает длины заголовков Ethernet и PPPoE.

9. Содержимое поля payload зависит от выполняемой стадии. В стадии Discovery оно содержит нуль или более атрибутов, называемых тегами (TAG). В стадии PPP Session оно содержит кадр PPP.

2.9.1. Функционирование PPPoE

PPPoE включает две стадии (рис. 2.36): *стадию обнаружения* (Discovery) и *стадию PPP-сессии* (PPP Session).

Когда узел хочет установить PPPoE-сессию, сначала он должен выполнить стадию Discovery для определения MAC-адреса узла противоположной стороны и установить PPPoE SESSION_ID. Хотя PPP определяет взаимодействие типа «точка-точка», стадия Discovery основана на клиент-серверном взаимодействии. В процессе обнаружения узел (клиент) определяет концентратор доступа (сервер). В зависимости от топологии в сети могут быть несколько концентраторов доступа, с которыми узел может взаимодействовать. Стадия Discovery позволяет узлу обнаружить все концентраторы доступа

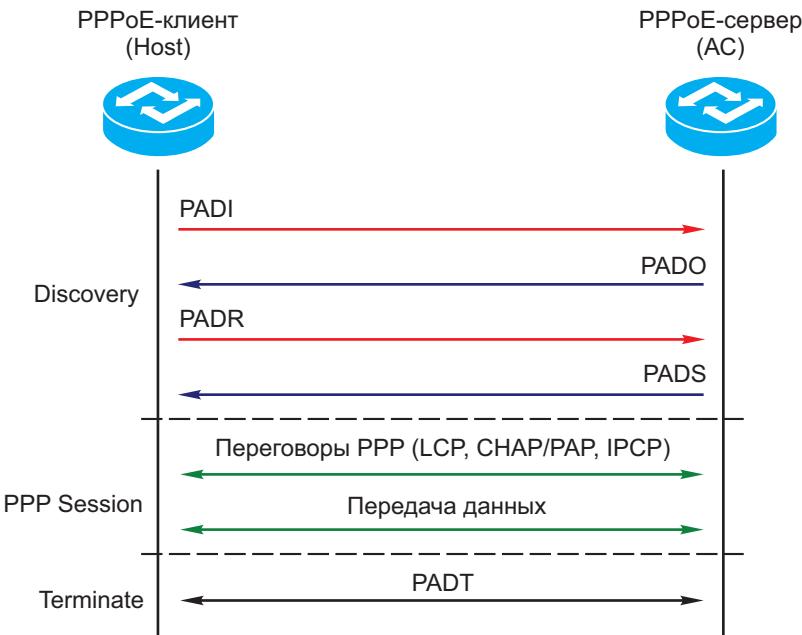


Рис. 2.36. Операции PPPoE

и выбрать один из них. При успешном завершении стадии Discovery узел и выбранный им концентратор доступа обладают информацией, позволяющей создать между ними соединение «точка-точка» через Ethernet.

Стадия Discovery не сохраняет состояние до тех пор, пока не установлена PPP-сессия. После установления PPP-сессии узел и концентратор доступа выделяют ресурсы для виртуального PPP-интерфейса.

Стадия Discovery

Стадия обнаружения (Discovery) состоит из четырех шагов. После завершения данной стадии оба участника знают SESSION_ID и MAC-адрес противоположной стороны канала, которые уникально определяют PPPoE-сессию. Шаги включают отправку узлом широковещательного пакета *Initiation* и получение от одного или более концентраторов доступа пакета *Offer*. Затем узел посыпает одноадресный пакет *Session Request*, и выбранный концентратор доступа отвечает пакетом *Confirmation*. После получения узлом пакета *Confirmation* он начинает стадию установления PPP-сессии. Значение поля ETHER_TYPE всех кадров Ethernet стадии Discovery равно 0x8863.

Содержимое поля данных пакетов PPPoE в стадии Discovery состоит из нуля или более атрибутов, которые в данном случае называются тегами (TAG).

Опишем подробнее выполнения стадии Discovery.

- Узел посылает пакет PPPoE Active Discovery Initiation (PADI) с широковещательным адресом, установленным в поле DESTINATION_ADDR

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	90:94:e4:f0:ca:4a	Broadcast	PPPoED	60	Active Discovery Initiation (PADI)
2	0.000108000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	77	Active Discovery Offer (PADO) AC-Name='linux_pppoe'
3	0.000417000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPPoED	60	Active Discovery Request (PADR)
4	0.000804000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	38	Active Discovery Session-confirmation (PADS)

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
 Ethernet II, Src: 90:94:e4:f0:ca:4a (90:94:e4:f0:ca:4a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 PPP-over-Ethernet Discovery
 0001 ... = Version: 1
 0001 = Type: 1
 Code: Active Discovery Initiation (PADI) (0x09)
 Session ID: 0x0000
 Payload Length: 12
 PPPoE Tags
 Host-Uniq: 180400000

Рис. 2.37. Пакет PADI

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	90:94:e4:f0:ca:4a	Broadcast	PPPoED	60	Active Discovery Initiation (PADI)
2	0.000108000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	77	Active Discovery Offer (PADO) AC-Name='linux_pppoe'
3	0.000417000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPPoED	60	Active Discovery Request (PADR)
4	0.000804000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	38	Active Discovery Session-confirmation (PADS)

Frame 2: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)
 Ethernet II, Src: Syskonne_9e:b2:b3 (00:00:5a:9e:b2:b3), Dst: 90:94:e4:f0:ca:4a (90:94:e4:f0:ca:4a)
 PPP-over-Ethernet Discovery
 0001 ... = Version: 1
 0001 = Type: 1
 Code: Active Discovery Offer (PADO) (0x07)
 Session ID: 0x0000
 Payload Length: 57
 PPPoE Tags
 AC-Name: linux_pppoe
 Service-Name: linux2
 AC-Cookie: 701a237837885f09ff33cce69f6868a7e34400000
 Host-Uniq: 180400000

Рис. 2.38. Пакет PAD

(рис. 2.37). Поле SESSION_ID должно иметь значение 0x0000. Пакет может содержать только один тег Service-Name, сообщающий о запрашиваемом клиентом сервисе и любое количество тегов других типов. Тег Host-Uniq используется узлом для уникальной ассоциации ответов концентратора доступа с запросами узла.

2. Когда концентратор доступа получает пакет PADI, который он может обработать, он отвечает пакетом PPPoE Active Discovery Offer (PADO) (рис. 2.38). Пакет PADO является одноадресным. Его поле DESTINATION_ADDR содержит уникальный MAC-адрес узла, отправившего пакет PADI. Поле SESSION_ID должно иметь значение 0x0000. Пакет PADO должен содержать тег AC-Name с именем концентратора доступа, тег Service-Name, идентичный тегу из пакета PADI, и любое количество тегов Service-Name, сообщающих о предлагаемых концентратором доступа сервисах. Если концентратор доступа не может обработать пакет PADI, он не отправляет пакет PADO.

3. Так как пакет PADI является широковещательным, узел может получить несколько пакетов PADO, из которых он должен выбрать один. Выбор может быть основан на AC-Name или предлагаемых сервисах. Затем узел отправляет пакет PPPoE Active Discovery Request (PADR) концентратору доступа, который он выбрал. Поле DESTINATION_ADDR пакета содержит уникальный MAC-адрес концентратора доступа, отправившего PADO. Поле SESSION_ID должно иметь значение 0x0000. Пакет может содержать только один тег Service-Name, сообщающий о запрашиваемом клиентом сервисе, и любое количество тегов других типов.

4. Когда концентратор доступа получает пакет PADR (рис. 2.39), он обрабатывает его и начинает PPP-сессию. Концентратор создает уникальный SESSION_ID для данной PPPoE-сессии и отвечает узлу пакетом PPPoE Active Discovery Session-confirmation (PADS) (рис. 2.40). Поле DESTINATION_ADDR пакета содержит уникальный MAC-адрес узла, отправившего PADR.

Стадия PPP-сессии

Стадия PPP-сессии (PPP Session) (рис. 2.41) может быть разделена на два шага: переговоров PPP и передачи данных PPP. В процессе переговоров PPP для PPPoE-сессии выполняются протоколы LCP, NCP и необходимые опциональные протоколы, например протоколы аутентификации. После завершения процесса переговоров начинается передача данных PPP. Данные PPP передаются, как любые другие вложенные данные. Все кадры Ethernet, передаваемые в течение PPPoE-сессии, являются одноадресными. Значение поля ETHER_TYPE в них равно 0x8864. Значение SESSION_ID, определенное на стадии обнаружения, не изменяется в течение всего периода работы данной PPPoE-сессии.

Напомним, что в процессе переговоров LCP стороны могут договориться о максимальном размере блока передаваемой по каналу дейтаграммы (MRU). Для PPPoE-сессии переговоры о MRU, размер которого больше 1492, вестись не должны. Это связано с тем, что максимальный размер поля данных Ethernet составляет 1500 байтов. Так как заголовок PPPoE равен 6 байтам

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	90:94:e4:f0:ca:4a	Broadcast	PPPoED	60	Active Discovery Initiation (PADT)
2	0.0001080000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	77	Active Discovery Offer (PADO) AC-Name= 'linux_pppoe'
3	0.0004170000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPPoED	60	Active Discovery Request (PADR)
4	0.0008040000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	38	Active Discovery Session-confirmation (PADS)
Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)						
Ethernet II, Src: 90:94:e4:f0:ca:4a (90:94:e4:f0:ca:4a), Dst: Syskonne_9e:b2:b3 (00:00:5a:9e:b2:b3)						
PPP-over-Ethernet Discovery						
0001 ... = Version: 1						
.... 0001 = Type: 1						
Code: Active Discovery Request (PADR) (0x19)						
Session ID: 0x00000						
Payload Length: 36						
PPPoE Tags						
Host-Unit: 18040000						
AC-cookie: 701aa237837885f09ff33cc669f6868a7e3440000						

Рис. 2.39. Текст PADR

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	90:94:e4:f0:ca:4a	Broadcast	PPPoED	60	Active Discovery Initiation (PADT)
2	0.0001080000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	77	Active Discovery Offer (PADO) AC-Name= 'linux_pppoe'
3	0.0004170000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPPoED	60	Active Discovery Request (PADR)
4	0.0008040000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPPoED	38	Active Discovery Session-confirmation (PADS)
Frame 4: 38 bytes on wire (304 bits), 38 bytes captured (304 bits)						
Ethernet II, Src: Syskonne_9e:b2:b3 (00:00:5a:9e:b2:b3), Dst: 90:94:e4:f0:ca:4a (90:94:e4:f0:ca:4a)						
PPP-over-Ethernet Discovery						
0001 ... = Version: 1						
.... 0001 = Type: 1						
Code: Active Discovery Session-confirmation (PADS) (0x65)						
Session ID: 0x0001						
Payload Length: 18						
PPPoE Tags						
Service-Name: linux2						
Host-Uniq: 18040000						

Рис. 2.40. Текст PADS

No.	Time	Source	Destination	Protocol	Length	Info
1	1.188589000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP LCP	60	Configuration Ack
8	3.011946000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP LCP	60	Configuration Request
9	3.012646000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP LCP	36	Configuration Ack
10	3.012671000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP CHAP	30	Echo Request
11	3.012687000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP LCP	48	Challenge (NAME='pptpd', VALUE=0x79dddf7f25b4af8e575bf52bbf5396d4)
12	3.013178000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP LCP	60	Echo Request
13	3.013210000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP LCP	60	Echo Reply
14	3.013582000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP LCP	30	Echo Reply
15	3.016210000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP CHAP	80	Response (NAME='test', VALUE=0x1655338bd9f3d3e36b4b618bc11891270060f5e)
16	3.018250000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP CHAP	85	Success (MESSAGE_=S=AE0EB73EC85F9F4CEC1F8B64A80A697800CDF2 M=Access)
17	3.018272000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP IPCP	32	Configuration Request
18	3.018887000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP IPCP	60	Configuration Request
19	3.019115000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP IPCP	60	Configuration Ack
20	3.019254000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP IPCP	44	Configuration Nak
21	3.019569000	90:94:e4:f0:ca:4a	Syskonne_9e:b2:b3	PPP IPCP	60	Configuration Request
22	3.020044000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP IPCP	44	Configuration Ack
23	3.020044000	Syskonne_9e:b2:b3	90:94:e4:f0:ca:4a	PPP IPCP	44	Configuration Ack
24	3.038777497	HewlettP_be:41:08	Broadcast	PPPoED	60	Active Discovery Initiation (PADT)
25	6.838861695	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	85	Active Discovery Offer (PADT) AC-Name='linux_ppoe'
26	6.839572944	HewlettP_be:41:08	AsustekC_40:d9:a5	PPPoED	70	Active Discovery Request (PADR)
27	6.840039977	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	46	Active Discovery Session-confirmation (PADS)
1469	22.385036696	HewlettP_be:41:08	AsustekC_40:d9:a5	PPPoED	60	Active Discovery Terminate (PADT)
1470	22.385108670	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	37	Active Discovery Terminate (PADT)

Рис. 2.41. Стадия PPP-сессии

No.	Time	Source	Destination	Protocol	Length	Info
24	3.038777497	HewlettP_be:41:08	Broadcast	PPPoED	60	Active Discovery Initiation (PADT)
25	6.838861695	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	85	Active Discovery Offer (PADT) AC-Name='linux_ppoe'
26	6.839572944	HewlettP_be:41:08	AsustekC_40:d9:a5	PPPoED	70	Active Discovery Request (PADR)
27	6.840039977	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	46	Active Discovery Session-confirmation (PADS)
1469	22.385036696	HewlettP_be:41:08	AsustekC_40:d9:a5	PPPoED	60	Active Discovery Terminate (PADT)
1470	22.385108670	AsustekC_40:d9:a5	HewlettP_be:41:08	PPPoED	37	Active Discovery Terminate (PADT)
▷ Frame 1469: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0						
▷ Ethernet II, Src: HewlettP_be:41:08 (a4:5d:36:be:41:08), Dst: AsustekC_40:d9:a5 (90:e6:ba:40:d9:a5)						
◀ PPP-over-Ethernet Discovery						
0001 ... = Version: 1						
... 0001 = Type: 1						
Code: active Discovery Terminate (PADT) (0xa7)						
Session ID: 0x0004						
Payload Length: 0						

Рис. 2.42. Пакет PADT

и поле Protocol PPP равно 2 байтам, то PPP MTU не может быть больше, чем 1492. Однако RFC 4638 «Accommodating a Maximum Transit Unit/Maximum Receive Unit (MTU/MRU) Greater Than 1492 in the Point-to-Point Protocol over Ethernet (PPPoE)» позволяет сторонам вести переговоры о MRU, размер которого больше 1492. Для этого взаимодействующие устройства должны поддерживать RFC 4638 и jumbo-фреймы Ethernet.

При использовании PPPoE инициированная клиентом PPP-сессия заканчивается на устройстве агрегирования провайдера, которое является и конечной точкой L2-соединения. Все действия по аутентификации клиентов, назначению им IP-адресов и других параметров выполняет устройство агрегирования провайдера. Так как в большинстве случаев абонентское устройство работает в режиме маршрутизатора, публичный IP-адрес назначается устройством провайдера его порту WAN. С помощью протокола DHCP абонентский маршрутизатор может присвоить IP-адреса подключенными к нему узлам. После присвоения IP-адресов маршруты к этим устройствам создаются в его таблице маршрутизации. В случае использования адресации IPv4 маршрутизатор может назначить подключенными к нему компьютерам частные адреса IPv4 и использовать трансляцию адресов (NAT). Маршрут к абонентскому маршрутизатору создается на агрегирующем устройстве провайдера. Когда все маршруты созданы, пользователь может получить доступ в Интернет.

Завершение PPPoE-сессии

В любое время после установления сессии узлом или концентратором доступа может быть отправлен пакет PPPoE Active Discovery Termination (PADT) (рис. 2.42), который сообщает, что PPPoE-сессия завершается. Поле DESTINATION_ADDR пакета содержит уникальный MAC-адрес получателя, поле SESSION_ID указывает номер сессии, которая завершается. Как только получен пакет PADT, передача трафика PPP для этой сессии должна прекратиться.

При нормальном режиме работы пакет PADT отправляется после того, как была завершена сессия PPP.

2.10. Передача PPP через ATM

В настоящее время протокол ATM используется на канальном уровне технологий семейства xDSL. Технологии Digital Subscriber Line (DSL) являются технологиями широкополосного доступа, которые используют провода традиционной телефонной сети для передачи данных. Наиболее широкое распространение среди них получила технология ADSL (рис. 2.43). Ее активно используют операторы связи для подключения домашних пользователей. На линиях ADSL применяются два альтернативных метода доступа: PPPoE и PPPoA.

Прежде чем приступить к описанию передачи кадров PPP через ATM, приведем краткое описание технологий ADSL и ATM.

2.10.1. Обзор технологии ADSL

ADSL (Asymmetric Digital Subscriber Loop, асимметричный цифровой абонентский контур) — технология семейства xDSL, обеспечивающая передачу данных через тот же локальный телефонный контур, по которому предоставляются услуги обычной аналоговой телефонии.

ADSL является технологией физического уровня модели OSI. Она отвечает за непосредственную транспортировку сигнала по витой телефонной паре между передатчиком и приемником. В упрощенном виде линия ADSL представляет собой соединение двух ADSL-модемов (клиентского и модема на узле провайдера услуг), которые подключены к концам витой пары телефонного кабеля. Модем на стороне клиента модулирует сигнал, прежде чем он будет передан в локальный телефонный контур. ADSL-модем на стороне провайдера демодулирует полученный сигнал. На канальном уровне ADSL использует протокол ATM (Asynchronous Transfer Mode, режим асинхронной передачи).

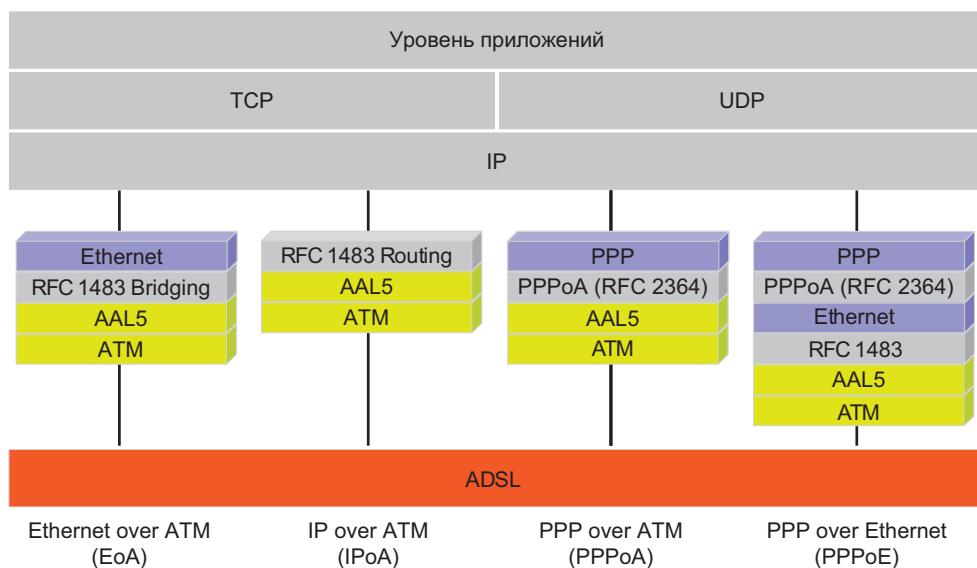


Рис. 2.43. Стек протоколов ADSL

На рис. 2.43 показаны протоколы, работающие на канале ADSL.

Аналогично технологии Ethernet, технология ADSL включает несколько реализаций. Они описаны в стандартах ANSI T1.413 и ITU-T G.992.x. Некоторые из стандартов приведены в табл. 2.2.

Таблица 2.2. Стандарты ADSL

Название технологии	Название стандарта	Год принятия стандарта	Максимальная скорость нисходящего потока	Максимальная скорость восходящего потока
ADSL	ANSI T1.413-1998 Issue 2	1998	6,144 Мбит/с	640 Кбит/с
ADSL (G.dmt)	ITU-T G.992.1	1999	6,144 Мбит/с (16 Мбит/с опционально)	640 Кбит/с (800 Кбит/с опционально)
ADSL (G.lite)	ITU-T G.992.2	1999	1,536 Мбит/с	512 Кбит/с
ADSL2 (G.dmt.bis)	ITU-T G.992.3	2002	8 Мбит/с (14,656 Мбит/с опционально)	800 Кбит/с (3,520 Мбит/с опционально)
ADSL2 (G.lite.bis)	ITU-T G.992.4	2002	1,536 Мбит/с	512 Кбит/с
READSL2	ITU-T G.992.3 Annex L	2003	8 Мбит/с (14,656 Мбит/с опционально)	800 Кбит/с (3,520 Мбит/с опционально)
ADSL2plus	ITU-T G.992.5	2003	16 Мбит/с (29,556 Мбит/с опционально)	800 Кбит/с (3,520 Мбит/с опционально)

В настройках абонентских ADSL-устройств можно включать или отключать поддержку соответствующих стандартов ADSL (рис. 2.44).

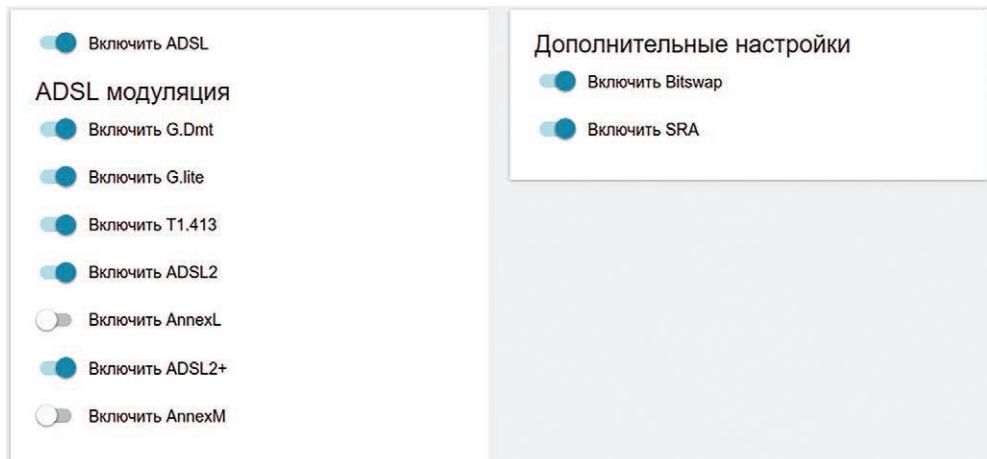


Рис. 2.44. Включение и отключение поддержки различных стандартов ADSL в настройках ADSL-маршрутизатора D-Link

2. Протокол PPP

Факторами, влияющими на скорость передачи данных через ADSL, являются состояние абонентской линии (например, диаметр жил, наличие кабельных отводов и т. п.) и ее протяженность. Затухание сигнала в линии повышается при увеличении ее длины и частоты сигнала, а снижается — при увеличении диаметра провода. Чем длиннее линия, тем меньшую скорость передачи она обеспечивает. Развитие технологии ADSL связано не только с повышением скорости передачи в направлении абонента, но и с увеличением расстояния, на котором будет достигнута максимальная скорость.

Следует отметить, что наряду с телефонной сетью общего пользования (ТфОП) ADSL также позволяет применять цифровые линии сетей ISDN. Использование технологии ADSL поверх обычной телефонной линии описано в Приложениях А (Annex A), а использование технологии ADSL поверх сети ISDN — в Приложениях В (Annex B) к стандартам ITU-T G.992.x. Абонентские устройства, предназначенные для работы через ТфОП, обозначаются как *Annex A*. Абонентские устройства, предназначенные для работы через ISDN, обозначаются как *Annex B*.

Преимуществом технологии ADSL (Annex B) перед ADSL (Annex A) является то, что она помимо двунаправленной передачи данных может быть использована для передачи телефонных сигналов ISDN, а также сигналов систем охранно-пожарной сигнализации. Тип используемой абонентской линии указывается в техническом описании оборудования ADSL, и этот параметр необходимо учитывать при выборе устройств для решения конкретной задачи.

На рис. 2.45 показаны эталонная модель ADSL Forum и основные компоненты сети ADSL. Все оборудование ADSL делится на два типа: ATU-R (ADSL Transmission Unit at the customer premises end) и ATU-C (ADSL Transmission Unit at the network end).

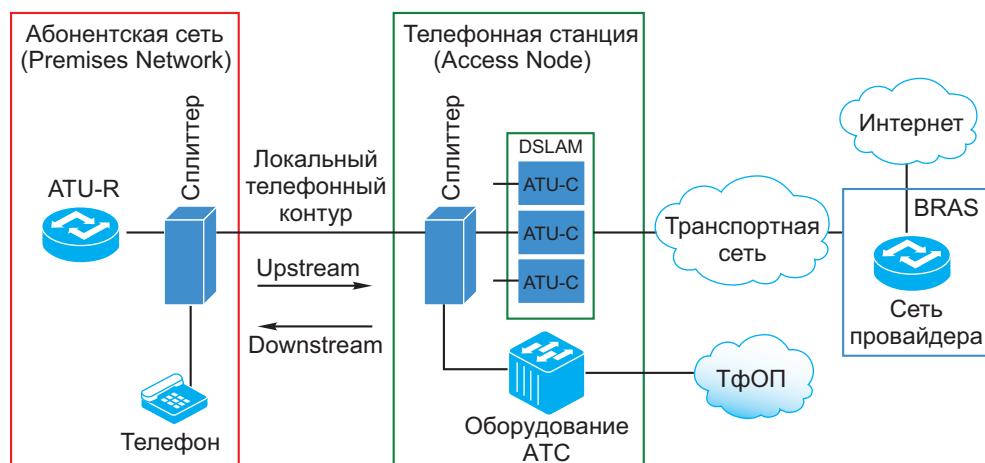


Рис. 2.45. Эталонная модель ADSL Forum

У клиентов устанавливаются ADSL-модемы, которые в терминологии ADSL Forum называются передающими абонентскими устройствами ADSL (ATU-R). Другое часто используемое название оборудования данного типа — абонентское оборудование (Customer Premises Equipment, CPE). В современных клиентских устройствах наряду с функцией ADSL-модема поддерживаются функции маршрутизации, межсетевого экрана, качества обслуживания (QoS), VLAN. Такие устройства называются ADSL-маршрутизаторами (рис. 2.46). Устройства пользователя подключаются к ADSL-маршрутизатору через Ethernet или по Wi-Fi.



Рис. 2.46. ADSL-маршрутизаторы компании D-Link

Когда ADSL-маршрутизатор получает IP-пакеты из локальной сети, на канальном уровне он помещает их в ячейки протокола ATM, а на физическом уровне модулирует сигнал и отправляет его по локальному телефонному контуру соответствующему ADSL-модему (ATU-C), расположенному в местном офисе телефонной компании. Оборудование ATU-C интегрировано в мультиплексор доступа DSL (DSL Access Multiplexer, DSLAM), который установлен на телефонной станции. DSLAM устанавливается до оборудования самой телефонной станции, поскольку он должен обеспечить мультиплексирование множества абонентских линий ADSL в одну высокоскоростную магистральную сеть (ATM или Ethernet), подключенную к сети провайдера услуг. Через сеть провайдера IP-пакеты достигают точки назначения. В сети провайдера услуг агрегацией трафика разных абонентов занимается *сервер широкополосного удаленного доступа* (Broadband Remote Access Server, BRAS). Он представляет собой пограничный IP-маршрутизатор для интеллектуального управления широкополосным доступом, который выполняет идентификацию клиентов, присвоение IP-адресов, управление параметрами трафика, маршрутизацию трафика к/от DSLAM.

2. Протокол PPP

Первое поколение DSLAM подключалось к BRAS через сеть ATM, так как в 1990-х годах ATM была основным транспортным средством высокоскоростных магистральных сетей. По мере развития Ethernet все больше провайдеров стали использовать эту технологию на своих транспортных магистралях (Metro Ethernet). Появились DSLAM, которые подключались к BRAS через Ethernet. DSLAM с uplink-портами Ethernet стали называться Ethernet DSLAM, или IP DSLAM (рис. 2.47). В простейшей реализации IP DSLAM функционирует как коммутатор L2. Помимо мультиплексирования абонентских линий, он позволяет провайдерам предоставлять клиентам различные сервисы, например IPTV, поддерживает функции управления трафиком, качества обслуживания (QoS), VLAN.



DAS-3216



DAS-3224



DAS-3248



DAS-4672

Рис. 2.47. IP DSLAM компании D-Link

В ADSL применяется метод разделения полосы пропускания медной телефонной линии на несколько частотных полос (рис. 2.48):

- диапазон от 0 до 4 кГц для передачи сигналов аналоговой телефонии (ТфОП);
- диапазон от 26 кГц до 138 кГц для передачи восходящего (Upstream) потока данных ADSL;
- диапазон от 138 кГц до 1,1 МГц для передачи нисходящего (Downstream) потока данных ADSL.

Доступная полоса пропускания ADSL распределена асимметрично между нисходящим и восходящим потоками данных. Эта асимметричность возникла из-за особенностей доступа в Интернет. Интернет-трафик характеризуется тем, что основной поток информации идет из сети к пользователю. Асимметричная цифровая абонентская линия обеспечивает оптимальное решение задачи доступа, так как ее скорость передачи данных в направлении

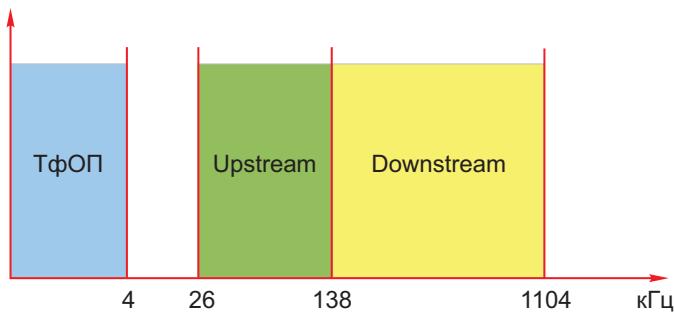


Рис. 2.48. Спектр ADSL

абонента значительно превышает скорость транспортировки данных от абонента в сеть. Кроме того, от пользователя в сеть сигнал передается на более низких частотах, чем те, которые используются при следовании сигнала в обратном направлении. Скорость нисходящего потока может быть от 32 Кбит/с до 29,5 Мбит/с в зависимости от используемого стандарта ADSL. Скорость восходящего потока лежит в диапазоне от 32 Кбит/с до 3,5 Мбит/с.

Для создания в полосе частот ADSL восходящих и нисходящих каналов используется частотное мультиплексирование, или **мультиплексирование с разделением по частоте** (*Frequency Division Multiplexing, FDM*) (рис. 2.49). Напомним, что в FDM широкая полоса пропускания физического канала F делится на n узких полос частот $f << F$, в каждой из которых создается логический канал. Размеры частотных полос f могут быть различными. Каждой взаимодействующей системе назначается отдельный поддиапазон частот (логический канал). Отправители могут посыпать сигналы одновременно. Передаваемые по разным логическим каналам сигналы накладываются на разные несущие и поэтому в частотной области не должны пересекаться. Для исключения влияния друг на друга сигналов, передаваемых по разным

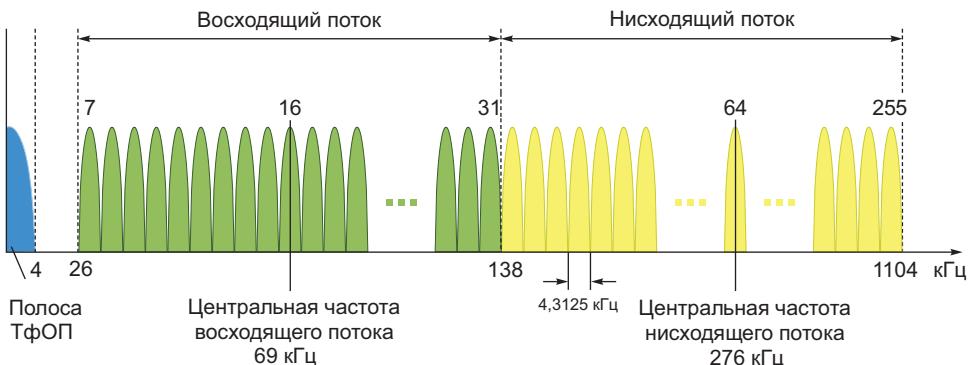


Рис. 2.49. Мультиплексирование с разделением по частоте (FDM)

логическим каналам, между ними формируются защитные полосы, служащие границами между каналами.

Метод, альтернативным FDM является **эхокомпенсация (Echo Cancelation, EC)**, которая позволяет сигналам восходящих и нисходящих потоков использовать одну полосу частот (рис. 2.50). При использовании эхокомпенсации спектр нисходящего канала становится шире. Сигнал передается на более низких частотах, что уменьшает его затухание. Теоретически, это позволяет повысить скорость передачи нисходящего потока данных на длинных телефонных линиях.

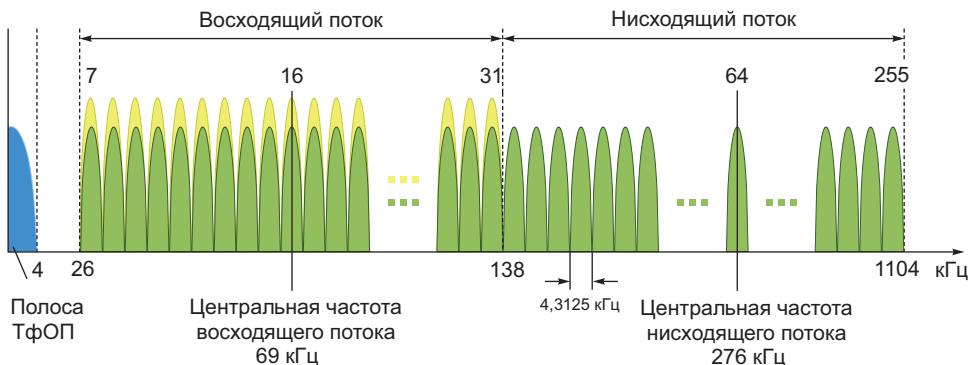


Рис. 2.50. Эхокомпенсация

В ADSL существуют две схемы модуляции: CAP (Carrierless Amplitude/Phase) и DMT (Discrete Multi Tone). Модуляция CAP не стандартизована. Модуляция DMT первоначально была определена в стандарте ANSI T1.413, а затем была перенесена в стандарт ITU G.992.1. Все современные устройства ADSL основаны на модуляции этого типа.

В DMT используется набор несущих частот, количество которых определяется числом каналов в полосе частот, занимаемой спектром DMT-сигнала. В большинстве случаев в полосе частот, занимаемой сигналом DMT, размещаются 256 частотных каналов. Каждый из этих каналов имеет ширину 4,3125 кГц и служит для организации независимой передачи данных. При подключении ADSL-маршрутизатора к DSLAM проводится проверка качества линии. После этого передатчик, исходя из уровня помех в частотном диапазоне сигнала DMT, для каждого из отдельных каналов выбирает подходящую схему модуляции. На частотных участках с малым уровнем шумов могут быть использованы методы с большими значениями спектральной плотности мощности, например QAM-64. На более зашумленных участках могут быть использованы простые алгоритмы модуляции. Использование такого принципа регулирования скорости передачи данных позволяет наиболее точно согласовывать параметры модулированного сигнала с параметрами линии, по которой он будет передаваться. При передаче данных

информация распределяется между независимыми каналами пропорционально их пропускной способности. Приемник выполняет операцию демультиплексирования и восстанавливает исходный информационный поток.



Рис. 2.51. Сплиттер DSL-39SP компании D-Link

ся по стандартной схеме, фильтр должен представлять собой пассивное устройство, не требующее питания.

Фильтр, устанавливаемый в помещении пользователя, представляет собой блок с тремя разъемами, служащими для подключения ADSL-маршрутизатора, телефонного оборудования и подсоединения к линии ADSL. Он разделяет спектр сигнала на стороне абонента.

Фильтр, устанавливаемый на телефонной станции, обеспечивает разделение ADSL и голосовых сигналов на другом конце абонентской линии. Далее аналоговый сигнал подается на коммутационное оборудование телефонной связи, а сигнал ADSL — на DSLAM. В связи с тем что современные DSLAM оборудованы портами ADSL со встроенными сплиттерами, нет необходимости устанавливать отдельные фильтры. Это позволяет провайдерам обеспечить интеграцию инфраструктуры ТфОП и ADSL с наименьшими затратами (рис. 2.52).



Рис. 2.52. Сплиттерный модуль ТфОП для IP DSLAM DAS-4672

2.10.2. Обзор технологии ATM

На канальном уровне ADSL использует технологию ATM. Технология ATM (*Asynchronous Transfer Mode, режим асинхронной передачи*) была разработана в начале 1990-х годов. Она основана на пересылке ячеек (cells) информации

2. Протокол PPP

фиксированной длины. Асинхронная передача означает, что ячейки пересыпаются только тогда, когда имеется какая-то информация, готовая к передаче. Размер каждой ячейки составляет 53 байта. Первые 5 байтов содержат заголовок, следующие 48 байтов — полезная нагрузка. Так как ячейки малы по размеру, они удобны для передачи голосовых и видеоданных, которые чувствительны к задержкам. Большие пакеты не будут приводить к длинным задержкам при пересылке фрагментов голосовой или видеинформации.

Сеть ATM состоит из двух основных компонентов — ATM-коммутатора и оконечного ATM-оборудования пользователя. ATM-коммутатор выполняет коммутацию ячеек, поступающих от других коммутаторов или оконечного оборудования пользователя, на основе информации, содержащейся в их заголовках. Оконечное оборудование пользователя инициирует отправку ячеек. Обычно сеть ATM состоит из множества коммутаторов, которые соединены между собой интерфейсами. Коммутаторы ATM поддерживают два типа интерфейсов:

- User-Network Interfaces (UNI) — интерфейс пользователя с сетью, который соединяет оконечное оборудование пользователя с ATM-коммутатором;
- Network-to-Network Interfaces (NNI) — интерфейс между сетями, который соединяет два ATM-коммутатора.

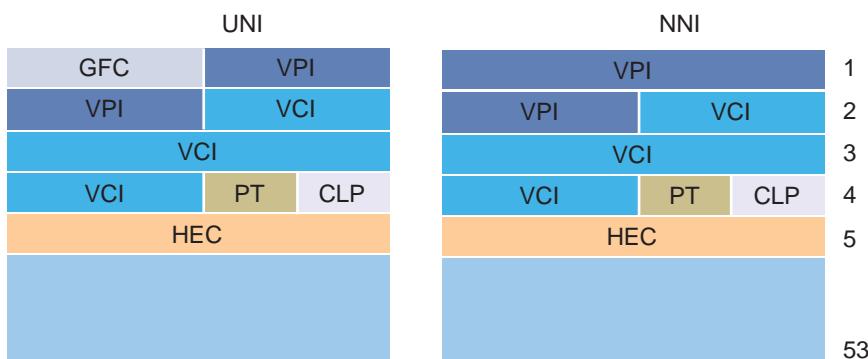


Рис. 2.53. Форматы ячеек UNI и NNI

Кратко опишем поля заголовков ячеек UNI и NNI (рис. 2.53):

- GFC (Generic Flow Control) — общее управление потоком, 4 бита. Используется только в UNI для управления трафиком и предотвращения перегрузок;
- VPI (Virtual Path Identifier) — идентификатор виртуального пути, используемый для маршрутизации ячейки. В UNI состоит из 8 битов, в NNI — из 12 битов;
- VCI (Virtual Channel Identifier) — идентификатор виртуального канала, используемый для маршрутизации ячейки, состоит из 16 битов;

- PT (Payload Type) — тип данных длиной 3 бита, используемый для обозначения типа полезной нагрузки в ячейке;
- CLP (Cell Loss Priority) — признак потери приоритета ячейки. Это один бит, который определяет возможность потери ячейкой своего приоритета. Если ячейку можно отбросить при перегрузке, этот бит установлен в 1;
- HEC (Header Error Control) — контрольная сумма заголовка длиной 8 битов.

Эталонная модель ATM состоит из трех основных уровней: физического, ATM и уровня адаптации ATM (ATM adaptation layer, AAL). Физический уровень ATM аналогичен физическому уровню модели OSI. Он отвечает за прием ячеек с уровня ATM, упаковку их в соответствующий формат для пересылки через физическую среду, установление границ ячеек, выделение ячеек из принятого битового потока и передачу их на уровень ATM. Уровень ATM отвечает за установление соединений и маршрутизацию по ним ячеек. Уровень адаптации ATM получает пакеты от вышестоящих протоколов и формирует последовательность ячеек, в которых размещаются передаваемые блоки данных. На стороне приемника данные из ячеек собираются в единый пакет. Эти действия выполняются на уровне AAL процессом, который называется сегментацией и обратной сборкой (segmentation and reassembly). Для поддержки различных типов трафика в технологии ATM определены несколько уровней адаптации. Для передачи пакетных данных определен уровень AAL5 (ATM Adaptation Layer 5, уровень адаптации ATM 5).

Формат кадра AAL5 показан на рис. 2.54. Роль заголовка у него исполняет концевик, содержащий сведения о длине (Length), а также 4-байтовый код CRC для обнаружения ошибок. Помимо полезной нагрузки (PDU Payload), в кадре AAL5 есть биты заполнения (Pad). Они дополняют общую длину, чтобы она была кратной 48 байтам. Таким образом, кадр можно будет поделить на целое число ячеек. Хранить адреса внутри кадра не нужно, так как в каждой ячейке имеются идентификаторы виртуального пути и канала.

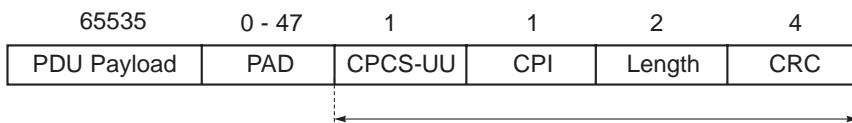


Рис. 2.54. Формат кадра AAL5

ATM является технологией с установлением соединения. Перед тем как начать передачу данных через сеть ATM, между двумя конечными точками должен быть установлен виртуальный канал. Правильная маршрутизация ячеек обеспечивается за счет того, что в их заголовках содержатся поля VPI и VCI. Идентификаторы VPI определяют *виртуальные пути* (virtual path), каждый из которых подразделяется на *виртуальные каналы* (virtual channel), идентифицируемые VCI (рис. 2.55). Пара идентификаторов VPI/VCI

2. Протокол PPP

позволяет определить путь передачи данных. При этом они имеют только локальное значение на данной линии связи и переназначаются на каждом устройстве, лежащем на пути передачи данных.



Рис. 2.55. Виртуальные пути и каналы

Виртуальные каналы ATM бывают трех видов:

- *Постоянный виртуальный канал (PVC, Permanent Virtual Circuit)*, который создается между двумя точками и существует в течение длительного времени, причем даже в отсутствие данных для передачи;
- *Коммутируемый виртуальный канал (SVC, Switched Virtual Circuit)*, который создается между двумя точками непосредственно перед передачей данных и разрывается после окончания сеанса связи;
- *Автоматически настраиваемый постоянный виртуальный канал (SPVC, Soft Permanent Virtual Circuit)*. Каналы SPVC по сути представляют собой каналы PVC, которые инициализируются по требованию в коммутаторах ATM.

В ADSL в большинстве случаев для установления соединения используются постоянные виртуальные каналы (PVC). Обычно каждому клиенту назначается один PVC, который коммутируется различными DSLAM до тех пор, пока не достигнет точки назначения. Провайдер предоставляет клиенту

The screenshot shows the configuration interface for an ATM connection. On the left, under 'Тип соединения' (Connection Type), 'PPPoE' is selected. Below it, 'Интерфейс' (Interface) is set to 'Добавить новый ATM PVC' (Add new ATM PVC). A toggle switch 'Включить соединение' (Enable connection) is turned on. The 'Имя соединения*' (Connection Name*) field contains 'PPPoE_100'. On the right, the 'ATM' section is expanded, showing fields for 'VPI (0-255)*' (set to 1), 'VCI (32 - 65535)*' (set to 35), 'Энкапсулация' (Encapsulation) set to 'LLC', and 'QoS класс' (QoS Class) set to 'UBR'.

Рис. 2.56. Окно настройки VPI/VCI на ADSL-маршрутизаторе D-Link

значения VPI и VCI, которые он должен самостоятельно ввести в настройках своего ADSL-маршрутизатора (рис. 2.56).

После того как рассмотрены технологии ADSL и AMT, давайте вернемся к вопросу передачи кадров PPP через ATM. Стандарт RFC 1483 (его заменил RFC 2684, но в технической документации все еще ссылаются на устаревший стандарт), описывает два метода инкапсуляции, позволяющих передавать трафик через сети ATM: routed Protocol Data Units (PDUs) over an ATM network и bridged Protocol Data Units (PDUs) over an ATM network. В первом методе трафик множества протоколов может передаваться через одно виртуальное соединение ATM (VC). Тип протокола, чей трафик передается в PDU, идентифицируется присоединенным в виде префикса заголовком IEEE 802.2 Logical Link Control (LLC). Во втором методе каждый виртуальный канал (VC) переносит трафик только одного определенного протокола. Другими словами, для трафика каждого протокола выделяется отдельный виртуальный канал. В обоих методах блоки данных протоколов должны инкапсулироваться в кадры AAL5.

Спецификация PPP Over AAL5 (PPPoA) была первым методом доступа, который определил DSL Forum. Она описывает инкапсуляцию кадров PPP в кадры AAL5 (RFC 2364) (рис. 2.57). Полезная нагрузка кадра AAL5 включает только поля *Протокол* (Protocol) и *Информация* (Information) кадра PPP. Поле протокола сообщает DSLAM, является ли полезная нагрузка IP-пакетом или пакетом другого протокола, например LCP. Принимающая сторона знает, что ячейки содержат информацию PPP, так как виртуальный канал ATM настраивается соответствующим образом. В кадре AAL5 механизмы формирования кадра PPP не требуются, всю работу выполняют ATM и AAL5. Код CRC кадра PPP также не нужен, поскольку AAL5 включает тот же самый код CRC.

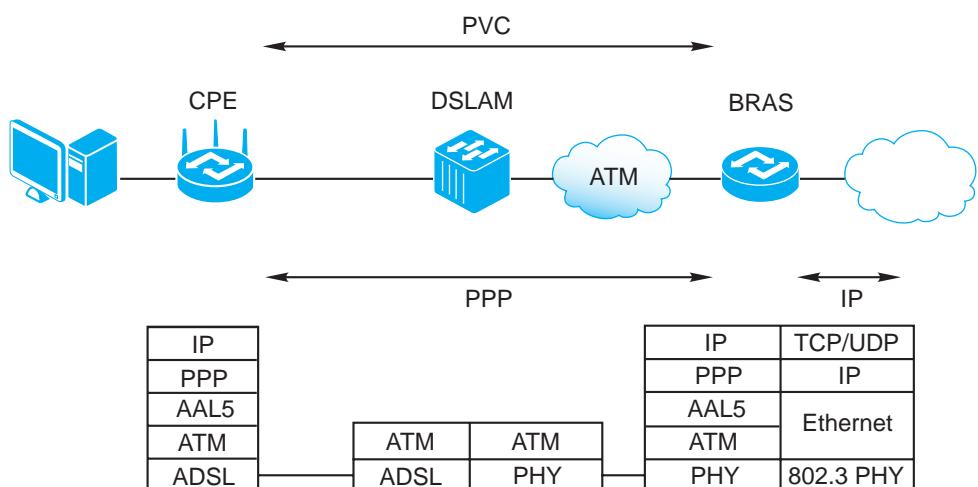


Рис. 2.57. Инкапсуляция PPPoA

2. Протокол PPP

PPPoE появился как альтернатива PPPoA и в настоящее время является превалирующим методом доступа. Для совместимости с абонентским оборудованием, поддерживающим только метод PPPoA, некоторые модели IPDSLAM поддерживают механизм трансляции PPPoA в PPPoE (рис. 2.58).

При использовании PPPoE на линии ADSL кадры PPP не помещаются напрямую в кадры AAL5. В кадры AAL5 с помощью методов инкапсуляции, описанных в RFC 1483, помещаются кадры Ethernet, содержащие кадры PPP (рис. 2.59).

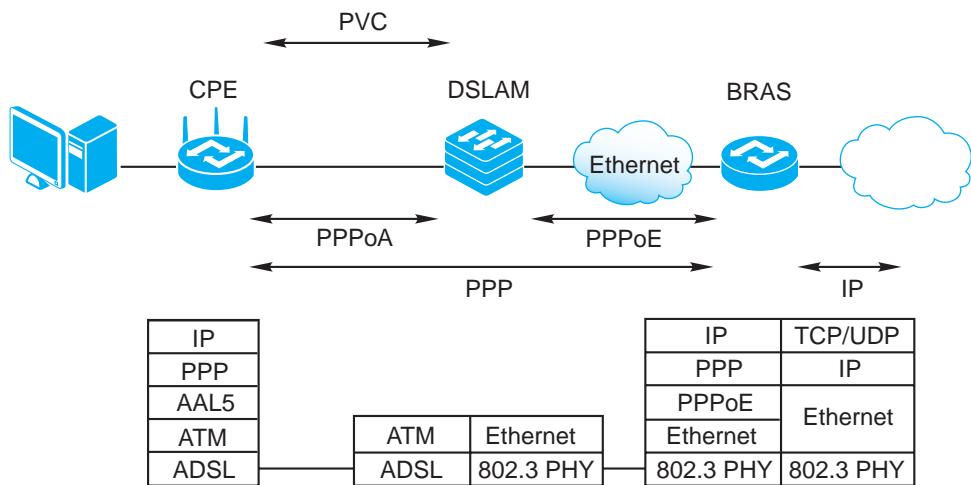


Рис. 2.58. Сеть с преобразованием PPPoA в PPPoE

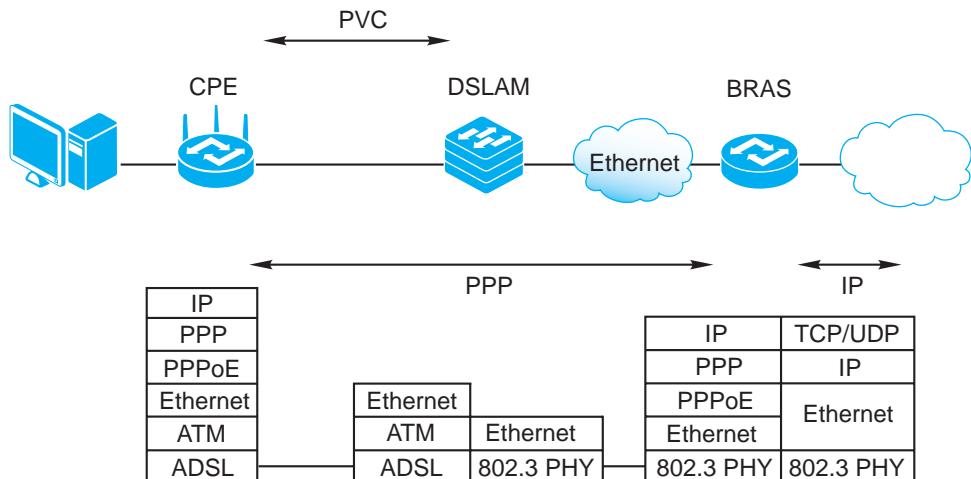


Рис. 2.59. Инкапсуляция PPPoE

2.11. Протокол PPTP

Протокол *Point-to-Point Tunneling Protocol* (PPTP) позволяет инкапсулировать кадры PPP в дейтаграммы IP и передавать их через IP-сеть. Он не вносит никаких изменений в протокол PPP, а только определяет механизм передачи его кадров. В протоколе используется клиент-серверная архитектура. Описан протокол PPTP в RFC 2637.

Для инкапсуляции кадров PPP в пакеты IP протокол PPTP использует модифицированную версию GRE (Generic Routing Encapsulation). Для установления, обслуживания и завершения туннеля между клиентом и сервером протокол PPTP организует управляющее соединение, использующее для работы порт TCP 1723. Для нормального функционирования PPTP необходимо, чтобы на стороне сервера было разрешено принимать входящие TCP-пакеты и запросы на установление соединений по данному порту.

Протокол PPTP может использоваться следующими способами:

1. Для подключения к сети провайдера с целью получения доступа в Интернет и других услуг.

2. Для подключения удаленных пользователей к корпоративной сети.

3. Для соединения удаленных локальных сетей одной организации.

Протокол PPTP выполняется между клиентом PPTP и сервером PPTP. В терминологии RFC 2637 клиент называется PPTP Access Concentrator (PAC). Клиентом PPTP может быть компьютер с установленным программным обеспечением PPTP или маршрутизатор с поддержкой этой функции. В терминологии RFC 2637 сервер называется PPTP Network Server (PNS). Он может быть реализован в виде программного обеспечения и работать на серверной платформе или являться функцией сетевого устройства, такого как маршрутизатор или межсетевой экран.

Протокол PPTP подразумевает, что между клиентом и сервером имеется IP-сеть. Соединение создается только между ними и никакие другие устройства не должны его поддерживать. При этом клиент может устанавливать соединения с разными серверами, а сервер может ассоциироваться с различными клиентами.

В PPTP существуют два компонента:

1) управляющее соединение между парой PAC-PNS, работающее через TCP;

2) IP-туннель, работающий между парой PAC-PNS, который используется для транспортировки инкапсулированных в GRE кадров PPP в течение пользовательских сессий.

Прежде чем установить туннель между клиентом и сервером, между ними должно быть установлено управляющее соединение. Управляющее соединение — это стандартная TCP-сессия, через которую PPTP передает контрольные и управляющие сообщения. Оно отвечает за установление, обслуживание и завершение PPP-сессий, передаваемых через туннель.

Установить управляющее соединение может как клиент, так и сервер. Для этого они обмениваются сообщениями *Start-Control-Connection-Request*

2. Протокол PPP

и *Start-Control-Connection-Reply*, которые содержат информацию о функциональных возможностях клиента и сервера. Как только управляющее соединение установлено, клиент или сервер могут инициировать создание PPP-сессий. Если создание сессии инициирует сервер, он отправляет клиенту сообщение *Outgoing-Call-Request*, на которое клиент отвечает сообщением *Outgoing-Call-Reply*. Если клиент инициирует создание сессии, он отправляет сообщение *Incoming-Call-Request*, на которое сервер отвечает сообщением *Incoming-Call-Reply*. Для проверки соединения между клиентом и сервером используются периодически отправляемые сообщения *Echo-Request* и *Echo-Reply*. Это позволяет своевременно обнаруживать неисправности. Для завершения туннеля сервер отправляет клиенту сообщение *Call-Clear-Request*, на которое он отвечает сообщением *Call-Disconnect-Notify*. Управляющее сообщение завершается после обмена сообщениями *Stop-Control-Connection-Request* и *Stop-Control-Connection-Reply*.

Все контрольные и управляющие сообщения передаются как пользовательские данные по установленному TCP-соединению между клиентом и сервером. Формат дейтаграммы показан на рис. 2.60.



Рис. 2.60. TCP-дейтаграмма, содержащая управляющее сообщение PPTP

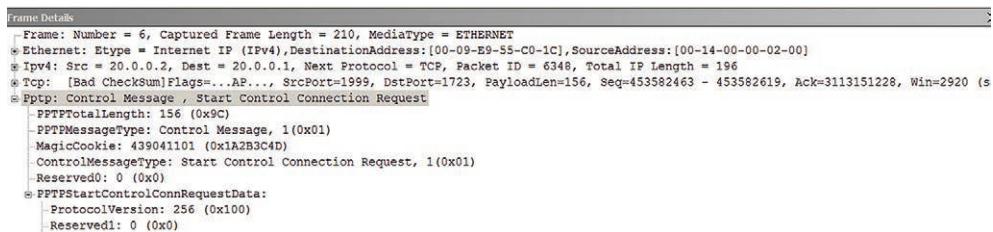


Рис. 2.61. Сообщение Start-Control-Connection-Request протокола PPTP

После того как клиент и сервер инициировали создание PPP-сессии, начинается процесс переговоров PPP и происходит последующая передача данных. В процессе переговоров PPP выполняются протоколы LCP, NCP и необходимые опциональные протоколы, например протоколы аутентификации. После завершения процесса переговоров начинается передача данных PPP. Все данные (управляющие и пользовательские) передаются в пакетах IP, содержащих кадры PPP, инкапсулированные в протокол GRE (рис. 2.61).

Протокол GRE (Generic Routing Encapsulation), описанный в RFC 1701 и 1702, позволяет инкапсулировать любой протокол сетевого уровня

в любой другой протокол сетевого уровня. Допустим, имеется пакет, который необходимо инкапсулировать и доставить некоторому получателю. Этот пакет будет являться содержимым GRE-пакета. Получившийся GRE-пакет затем инкапсулируется в пакет некоторого другого протокола и отправляется. Назовем этот внешний протокол протоколом доставки. Результирующий пакет имеет формат, показанный на рис. 2.62.



Рис. 2.62. Формат GRE-пакета

В заголовке GRE указывается тип инкапсулированного протокола. Если содержимым GRE-пакета является PPP, то тип протокола должен быть 0x880B. Туннель идентифицируется значением, указываемым в поле Call ID заголовка GRE.

Пакет GRE, содержащий кадр PPP (рис. 2.63), инкапсулируется в пакет IP. В качестве адресов источника и приемника в заголовке IP указываются IP-адрес клиента и IP-адрес сервера. После добавления заголовка IP пакет помещается в кадр канального уровня и передается по сети.

```
Frame Details
Frame: Number = 15, Captured Frame Length = 60, MediaType = ETHERNET
Ethernet: Etype = Internet IP (IPv4), DestinationAddress:[00-09-E9-55-C0-1C],SourceAddress:[00-14-00-00-02-00]
Ipv4: Src = 20.0.0.2, Dest = 20.0.0.1, Next Protocol = GRE, Packet ID = 6354, Total IP Length = 46
Gre: Protocol = PPP, Flags = ..KS..... Version 1 , Length = 0xE , CallID = 0x18
  flags: ..KS..... Version 1
  NextProtocol: PPP
  PayloadLength: 14 (0xE)
  CallID: 24 (0x18)
  SequenceNumber: 2 (0x2)
  Fpp: LCP, Link Control Protocol
  Lcp: Configure-Request, ID = 1, Length = 10
```

Рис. 2.63. GRE-пакет, содержащий кадр PPP

PPTP-сервер может быть размещен на серверной платформе, находящейся во внутренней сети за межсетевым экраном или маршрутизатором. В этом случае в качестве IP-адреса сервера указывается IP-адрес WAN-интерфейса маршрутизатора или межсетевого экрана. На устройствах, находящихся на границе сети, необходимо сделать настройки, которые позволят перенаправлять пакеты с номером порта TCP 1723 на PPTP-сервер (рис. 2.64).

В том случае, когда программное обеспечение клиента PPTP установлено на компьютер локальной сети, на маршрутизаторе, к которому он подключен, надо активировать функцию *PPTP pass through* (проброс PPTP). Эта функция разрешает маршрутизатору пропускать PPTP-трафик, позволяя клиентам из локальной сети устанавливать соответствующие соединения с удаленными сетями.

При подключении PPTP инициированная клиентом PPP-сессия может заканчиваться либо на стороне провайдера, либо на стороне удаленной сети.

2. Протокол PPP

Сессия заканчивается на стороне провайдера, когда протокол PPTP используется провайдером в качестве метода доступа. В этом случае PPP-соединение контролируется провайдером: он аутентифицирует клиентов, назначает им IP-адреса и другие параметры, обеспечивает доступ в Интернет.

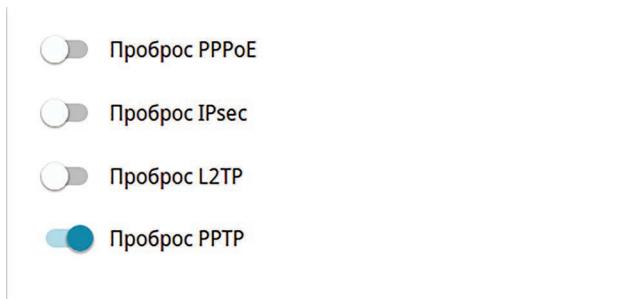


Рис. 2.64. Настройка функции проброса PPTP на маршрутизаторе D-Link

Сессия PPP заканчивается на стороне удаленной сети в том случае, если пользователь подключается к удаленной сети или соединяются локальные сети одной организации. Для установления туннеля PPTP (рис. 2.65) всем сторонам необходимо иметь подключение к Интернету. В этом случае PPP-соединение контролирует администратор сети организации. Аутентификация удаленных клиентов выполняется на серверах, принадлежащих компаниям, тип шифрования соединения выбирается в соответствии с политиками безопасности, определенными в ней (рис. 2.66). Также удаленным клиентам могут назначаться IP-адреса и другие параметры.

Описанные сетевые модели не исключают друг друга и могут использоваться одновременно. Для того чтобы установить туннель, пользователь должен подключиться к серверу доступа провайдера и установить с ним PPP-соединение. Это соединение может использоваться для доступа в Интернет и получения сервисов, предоставляемых провайдером. Туннель между пользователем и удаленной сетью создается поверх установленного PPP-соединения с провайдером. Настраиваются и функционируют эти соединения независимо друг от друга. При настройке подключения к Интернету вводятся параметры, предоставленные провайдером. При настройке удаленного подключения указываются параметры, необходимые для получения доступа к удаленной сети.

Методы аутентификации PPTP наследуют от протокола PPP: PAP, CHAP, MS-CHAP, EAP. Для шифрования кадров PPP используется протокол MPPE. Шифрование, так же как и аутентификация, является необязательной опцией. При PPTP-подключении к провайдеру шифрование кадров обычно не используется, поскольку за пределами соединения клиент-провайдер данные

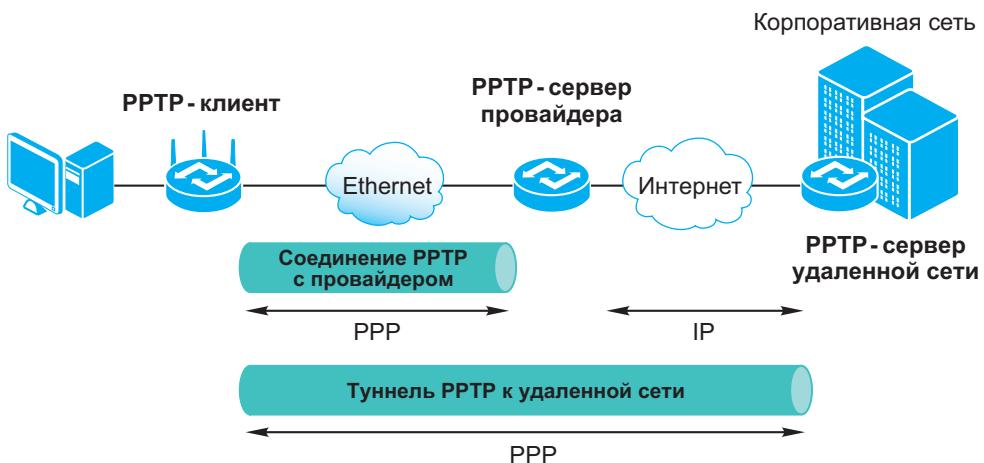


Рис. 2.65. Туннели PPTP

Основные		Все настройки	
Тип соединения	PPTP	PPP	
<input checked="" type="checkbox"/> Включить соединение	<input type="checkbox"/> Без авторизации		
Имя соединения*	Имя пользователя*		
PPTP_50	test		
	Пароль*	Пароль*	
	****	****	
	Адрес VPN-сервера*	Адрес VPN-сервера*	
	test.vpn.local	test.vpn.local	
	MTU*	MTU*	
	1456	1456	
	Протокол аутентификации	Протокол аутентификации	
	AUTO	AUTO	
	Протокол шифрования	Протокол шифрования	
	Без шифрования	Без шифрования	
	Без шифрования		
	MPPE 40 128 bit		
	MPPE 40 bit		
	MPPE 128 bit		
	3U		

Рис. 2.66. Настройка шифрования PPTP на маршрутизаторе D-Link

2. Протокол PPP

будут передаваться в открытом виде, а дополнительные операции по шифрованию будут снижать производительность. При создании PPTP-подключения к удаленной сети рекомендуется включать шифрование, так как по этому соединению могут передаваться конфиденциальные данные.

Заголовок канального уровня	Заголовок IP	Заголовок GRE	Заголовок PPP	Зашифрованная полезная нагрузка PPP (IP-дейтаграмма)	Концевик канального уровня
-----------------------------	--------------	---------------	---------------	--	----------------------------

Рис. 2.67. Туннелированные зашифрованные данные PPTP

Следует отметить, что шифруется не весь кадр, а полезная нагрузка PPP, т. е. IP-пакет, первоначально инкапсулированный в кадр PPP (рис. 2.67).

Как уже упоминалось ранее, протокол MPPE имеет известные принципиальные уязвимости, позволяющие взломать его. Поэтому он обеспечивает слабую защиту и его не рекомендуется использовать для шифрования критически важных данных.

2.12. Протокол L2TP

Протокол Layer Two Tunneling Protocol (L2TP) является комбинацией протоколов PPTP и Layer 2 Forwarding (L2F), технологии, предложенной компанией Cisco Systems. IETF объединила в протоколе L2TP лучшие характеристики этих протоколов.

В настоящее время существуют две реализации протокола L2TP — L2TP версии 2 (RFC 2661) и L2TP версии 3 (RFC 3931).

L2TP версии 2 (L2TPv2) определяет инкапсуляцию кадров PPP и их передачу через IP-сеть. L2TP версии 3 (L2TPv3) определяет инкапсуляцию кадров любых протоколов L2 (PPP, Ethernet, ATM и т.д.) и их передачу через IP-сеть. В данном разделе будет рассмотрена работа L2TPv2.

Протокол L2TP расширяет возможности протокола PPP. Он позволяет конечным точкам соединения канального уровня и сессии PPP находиться на разных устройствах, соединенных между собой IP-сетью. Другими словами, когда пользователи подключаются к концентратору доступа (например, к IPDSLAM при использовании ADSL), сессия PPP не обязательно должна на нем заканчиваться. Она может закончиться на другом устройстве, подключение к которому выполняется через разделяемую сетевую инфраструктуру, такую как Интернет.

L2TP может использоваться несколькими способами:

1. Для подключения к сети провайдера с целью получения доступа в Интернет и других услуг.
2. Для подключения удаленных пользователей к корпоративной сети.
3. Для соединения удаленных локальных сетей одной организации.

Терминология L2TP:

L2TP Access Concentrator (*LAC*, концентратор доступа *L2TP*) — это узел, который действует как одна из конечных точек туннеля и является стороной, противоположной *LNS* (*L2TP Network Server*). Он расположен между *LNS* и удаленной системой и перенаправляет между ними пакеты. Пакеты, посылаемые от *LAC* к *LNS*, туннелируются по протоколу *L2TP*. Соединение между *LAC* и удаленной системой является либо локальным (см. *Client LAC*), либо выполняется по протоколу *PPP*.

Client LAC (клиент *LAC*) — это узел, на котором установлено программное обеспечение *L2TP*. Он может непосредственно участвовать в туннелировании и подключаться к *LNS* без использования отдельного *LAC*. Узел должен быть подключен к Интернету.

L2TP Network Server (*LNS*, сетевой сервер *L2TP*) — это узел, который действует как одна из конечных точек туннеля и является стороной, противоположной *L2TP Access Concentrator* (*LAC*). *LNS* является логической конечной точкой *PPP*-сессии, которая туннелируется от удаленной системы с помощью *LAC*.

Remote System (удаленная система) — это компьютер или маршрутизатор, подключенный к удаленной сети доступа, который или инициирует, или получает вызовы.

Network Access Server (*NAS*, сервер сетевого доступа) — это устройство, обеспечивающее пользователям доступ к локальной сети через удаленную сеть доступа, такую как телефонная сеть общего пользования. *NAS* может выполнять функции *LAC*, *LNS* или их обоих.

Call (вызов) — это соединение (или попытка соединения) между удаленной системой и *LAC*. Например, через телефонную сеть (*ADSL*, *Dial-up*, *ISDN*). Вызов (входящий или исходящий), успешно установленный между удаленной системой и *LAC*, приводит к установлению *L2TP*-сессии внутри туннеля, ранее установленного между *LAC* и *LNS*.

Tunnel (туннель) — туннель существует между парой *LAC-LNS*. Он состоит из управляющего соединения и 0 или более *L2TP*-сессий. По туннелю передаются инкапсулированные кадры *PPP* и управляющие сообщения.

Session (сессия) — *L2TP*-сессия создается между *LAC* и *LNS*, когда между удаленной системой и *LNS* установлено *PPP*-соединение. Кадры, относящиеся к *PPP*-соединению, передаются между *LAC* и *LNS* через туннель. *LNS* и *LAC* поддерживают состояние для каждого вызова, который инициирован или на который ответил *LAC*.

Протокол *L2TP* определяет создание туннелей между *LAC* и *LNS* и последующую инкапсуляцию туннелируемых *PPP*-сессий. На рис. 2.68 показаны типичные случаи использования *L2TP*.

В первом случае удаленная система (на рисунке — *ADSL*-маршрутизатор) инициирует *PPP*-соединение к *LAC*, роль которого исполняет *BRAS*, расположенный в сети провайдера. Затем *LAC* туннелирует *PPP*-соединение через Интернет к *LNS*, роль которого исполняет маршрутизатор или межсетевой экран, стоящий на границе корпоративной и публичной сети. Удаленная система получает IP-адреса и другие параметры из корпоративной сети

2. Протокол PPP

с целью получения доступа к ее внутренним ресурсам. Аутентификацию, авторизацию и регистрацию пользователя, подключившегося к корпоративной сети, выполняет ее администратор. LNS может поддерживать функцию NAT для трансляции частных IP-адресов в публичные.

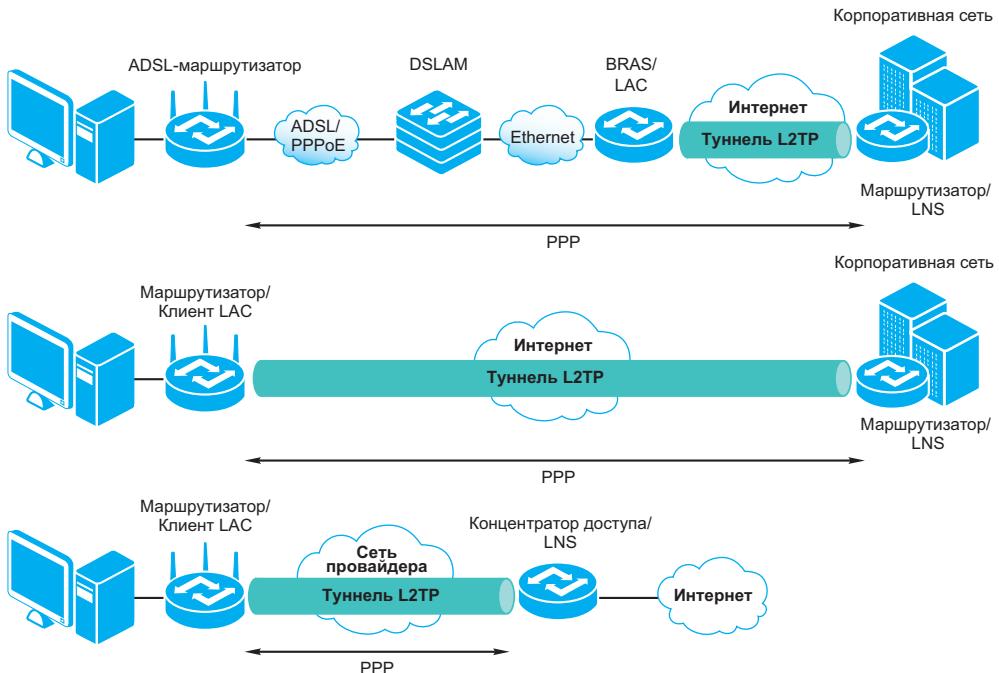


Рис. 2.68. Типичные топологии L2TP

Описанный сценарий может использоваться, когда абонентские устройства удаленных пользователей/сетей не поддерживают функционал клиента L2TP (рис. 2.69). Провайдеры могут оказывать таким абонентам услуги по созданию VPN-сетей на основе своих магистральных сетей. При этом абонентское устройство пользователя может подключаться к провайдеру любым способом.

В установлении туннеля к LNS может участвовать клиент LAC. Клиентом LAC может быть компьютер с установленным программным обеспечением L2TP, маршрутизатор или межсетевой экран с поддержкой функции клиента L2TP. Этую функцию поддерживают маршрутизаторы D-Link серий DIR-xxx, DSR-xxx, межсетевые экраны DFL-xxx. Оборудование с клиентом L2TP должно иметь подключение к Интернету. В этом случае создается виртуальное PPP-соединение, и клиент напрямую устанавливает туннель к LNS (рис. 2.70). Аутентификацию, авторизацию и регистрацию пользователя, как и в предыдущем случае, будет выполнять администратор корпоративной сети.

Технологии TCP/IP в современных компьютерных сетях

Основные

Все настройки

Тип соединения
L2TP

Без авторизации

Имя пользователя*
test

Пароль*

Адрес VPN-сервера*
l2tp vpn.local

ПРИМЕНЕНИТЬ

Рис. 2.69. Настройка клиента L2TP на маршрутизаторе D-Link

D-Link®

Unified Services Router - DSR-500

Logged in as: admin (ADMIN) | Language: English [US] [Logout](#)

Serial: S3B01G9000020 | Firmware: 3.07_RU

[Wizard](#) [System Search...](#) [?](#) [e](#)

Status Network VPN Security Maintenance

Security > Authentication > User Database > Users

Operation Succeeded

Get User DB Groups Users

This page shows a list of available users in the system. A user can add, delete and edit the users also. This page can also be used for setting policies on users.

Users List

Show 10 entries [Right click on record to get more options]

User Name	Group Name	Login Status
admin	ADMIN	Enabled (LAN) Enabled (WAN)
Client1	ClientL2TP	Enabled (LAN) Enabled (WAN)
guest	GUEST	Disabled (LAN) Disabled (WAN)

Showing 1 to 3 of 3 entries

[<] First [<] Previous 1 Next [>] Last [>]

Add New User

Рис. 2.70. Локальная база данных пользователей на маршрутизаторе DSR-500

Некоторые провайдеры используют L2TP в качестве метода доступа. Клиент L2TP подключается к LNS сети провайдера. Все функции по назначению IP-адресов и аутентификации пользователей выполняет провайдер.

LAC (клиент LAC) устанавливает L2TP-соединение с LNS на основе информации об имени пользователя, передаваемого в кадрах PPP. Он может создать множество туннелей к различным LNS для изоляции потоков данных.

LNS может быть точкой завершения множества туннелей от различных LAC. Он выполняет аутентификацию подключаемых к нему LAC с помощью локальной базы данных или сервера аутентификации.

Сообщения L2TP

Протокол L2TP (рис. 2.71) использует два типа сообщений:

- *Управляющие сообщения*. Они служат для установления, обслуживания и завершения туннелей и вызовов. Контрольные сообщения используют надежный управляющий канал внутри L2TP, чтобы гарантировать доставку. В IP-сетях управляющие сообщения L2TP инкапсулируются в дейтаграммы протокола UDP;

- *Сообщения данных*. Они используются для инкапсуляции кадров PPP, передаваемых через туннель. Сообщения данных передаются через ненадежный канал. Когда пакеты теряются, эти сообщения повторно не передаются. Кадры PPP инкапсулируются в пакет L2TP, а затем в пакет протокола, слушающего в качестве транспорта. В IP-сетях пакеты L2TP инкапсулируются в дейтаграммы протокола UDP.

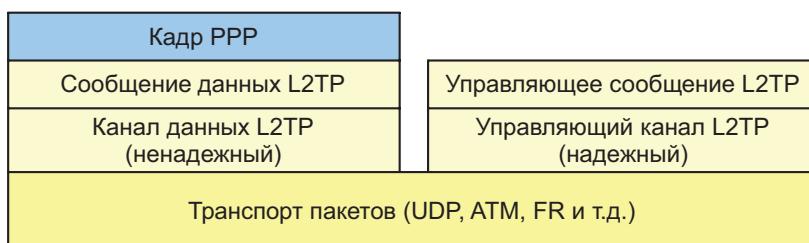


Рис. 2.71. Структура протокола L2TP

Управляющие сообщения и сообщения данных используют одинаковый формат заголовка пакета L2TP (рис. 2.72). Заголовок содержит идентификатор туннеля (Tunnel ID) и идентификатор сессии (Session ID). Идентификатор туннеля определяет управляющее соединение. Туннелям L2TP присваиваются идентификаторы, которые имеют только локальное значение, т. -е. один и тот же туннель на каждом из концов будет иметь различные значения Tunnel ID. В сообщениях L2TP в поле Tunnel ID указывается идентификатор

- Layer 2 Tunneling Protocol
 - Packet Type: Control Message Tunnel Id=6 Session Id=1
 - Length: 28
 - Tunnel ID: 6
 - Session ID: 1
 - Ns: 1
 - Nr: 3
 - Control Message AVP
 - Assigned Session AVP

Рис. 2.72. Заголовок пакета L2TP

получателя. Стороны выбирают Tunnel ID и обмениваются ими в процессе создания туннеля.

Идентификатор сессии определяет сессию внутри туннеля. Сессиям L2TP присваиваются идентификаторы, имеющие только локальное значение. Одна и та же сессия на разных концах туннеля будет иметь разные идентификаторы. В сообщениях L2TP в поле Session ID указывается идентификатор получателя. Стороны выбирают Session ID и обмениваются ими в процессе создания сессии.

Пакеты с одинаковым идентификатором туннеля, но разными идентификаторами сессии передаются через один туннель.

В IP-сетях пакеты L2TP инкапсулируются в UDP-дейтаграммы (рис. 2.73). Хорошо известный номер порта UDP для L2TP — 1701. Этот номер порта используется в начальной стадии установки туннеля. Инициатор L2TP-туннеля выбирает доступный UDP-порт источника (который может отличаться от 1701) и отправляет пакет получателю на порт 1701. Получатель выбирает свободный порт в своей системе (который может отличаться от 1701) и отправляет ответ инициатору на его произвольно выбранный UDP-порт. После того как адреса и порты инициатора и получателя установлены, они остаются постоянными в течение всего времени жизни канала. Так как получатель может выбрать произвольный порт источника (в отличие от порта назначения в пакете, инициирующем туннель, который должен быть 1701), это может вызвать проблемы, связанные с прохождением NAT.

UDP-дейтаграмма, содержащая пакет L2TP, помещается в IP-пакет, адресами источника и назначения которого являются IP-адреса LAC и LNS. Этот IP-пакет помещается в кадр канального уровня.

- Frame 124: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)
- Ethernet II, Src: 90:94:e4:f0:ca:4a (90:94:e4:f0:ca:4a), Dst: a0:ab:1b:f0:ef:c8 (a0:ab:1b:f0:ef:c8)
- Internet Protocol Version 4, Src: 172.16.5.2 (172.16.5.2), Dst: 172.16.5.1 (172.16.5.1)
- User Datagram Protocol, Src Port: 39100 (39100), Dst Port: 12f (1701)
- Layer 2 Tunneling Protocol
- Point-to-Point Protocol
- PPP Link Control Protocol

Рис. 2.73. Инкапсулированный пакет L2TP

Туннели и сессии

Процесс туннелирования PPP-сессий с помощью протокола L2TP состоит из двух шагов (рис. 2.74):

1. Установление управляющего соединения для туннеля;
2. Установление сессии путем отправки запросов входящих и исходящих вызовов.

Туннель и соответствующее управляющее соединение должны быть установлены до того, как будут инициированы входящие и исходящие вызовы. Сессия L2TP должна быть установлена до того, как туннель начнет передавать кадры PPP.

Между парой LAC и LNS может быть установлено множество туннелей. Туннель состоит из управляющего соединения и одной или нескольких сессий. L2TP-сессия может быть установлена только после успешного создания туннеля. Сессия соответствует одному потоку данных PPP между LAC и LNS.

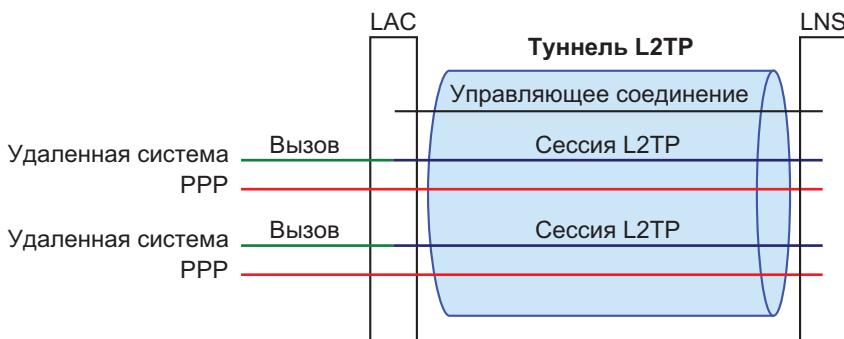


Рис. 2.74. Туннелирование PPP

Через туннель передаются и управляющие сообщения, и сообщения данных. Для проверки соединения LAC и LNS периодически отправляют друг другу сообщения Hello. Если от противоположной стороны в течение определенного периода времени не получен ответ, туннель разрывается.

Установление управляющего соединения

Управляющее соединение является начальным соединением, которое должно быть установлено до того, как между LAC и LNS будут установлены сессии. Установление управляющего соединения (рис. 2.75) включает обеспечение безопасности идентификаций участников и т. п.

Установление управляющего соединения начинается с обмена LAC и LNS управляющими сообщениями. Инициировать процесс может и LAC, и LNS. Инициатор установления туннеля отправляет сообщение *Start-Control-Connection-Request* (SCCRQ). В ответ на него противоположная сторона отправляет сообщение *Start-Control-Connection-Reply* (SCCRP). Оно показывает,

Source	Destination	Protocol	Length	Info
172.16.5.2	172.16.5.1	L2TP	179	Control Message - SCCRQ (tunne] id=0, session id=0)
172.16.5.2	172.16.5.1	L2TP	179	Control Message - SCCRQ (tunne] id=0, session id=0)
172.16.5.1	172.16.5.2	L2TP	195	Control Message - SCCRQ (tunne] id=53844, session id=0)
172.16.5.1	172.16.5.2	L2TP	195	Control Message - SCCRQ (tunne] id=53844, session id=0)
172.16.5.2	172.16.5.1	L2TP	62	Control Message - SCCCN (tunne] id=16869, session id=0)

Рис. 2.75. Установление управляющего соединения

Source	Destination	Protocol	Length	Info
172.16.5.2	172.16.5.1	L2TP	90	Control Message - ICRQ (tunne] id=16869, session id=0)
172.16.5.2	172.16.5.1	L2TP	90	Control Message - ICRQ (tunne] id=16869, session id=0)
172.16.5.1	172.16.5.2	L2TP	70	Control Message - ICRP (tunne] id=53844, session id=51038)
172.16.5.1	172.16.5.2	L2TP	70	Control Message - ICRP (tunne] id=53844, session id=51038)
172.16.5.2	172.16.5.1	L2TP	82	Control Message - ICCN (tunne] id=16869, session id=6208)
172.16.5.2	172.16.5.1	L2TP	82	Control Message - ICCN (tunne] id=16869, session id=6208)

Рис. 2.76. Установление входящего вызова

что сообщение *SCCRQ* принято и установка туннеля может продолжаться. В ответ на сообщение *SCCRP* отправляется сообщение *Start-Control-Connection-Connected* (*SCCCN*). Оно завершает процесс установления туннеля.

Аутентификация туннеля

В процессе установления управляющего соединения L2TP выполняет простую, не обязательную, CHAP-подобную аутентификацию туннеля. Если LAC или LNS хочет аутентифицировать другого участника, в сообщение *SCCRQ* или *SCCRP* включается атрибут *Challenge AVP*. Если получено сообщение *SCCRQ* или *SCCRP* с *Challenge AVP*, то *Challenge Response AVP* должен быть отправлен в следующем сообщении *SCCRP* или *SCCRN* соответственно. Если полученный ответ не соответствует ожидаемому, установление туннеля должно быть прекращено. Аутентификация выполняется с помощью разделяемого между LAC и LNS секрета.

Установление L2TP-сессии

После успешного установления управляющего соединения могут создаваться отдельные сессии. Каждая сессия соответствует одному потоку PPP между LAC и LNS. В отличие от установления управляющего соединения, установление сессии является направленным применительно к LAC и LNS. LAC запрашивает LNS принять сессию для входящего вызова (*incoming call*), LNS запрашивает LAC принять сессию для исходящего вызова (*outgoing call*).

Для установления входящего вызова (рис. 2.76), который инициирует LAC, между ним и LNS выполняется обмен тремя сообщениями. Когда входящий вызов определен, LAC отправляет LNS сообщение *Incoming-Call-Request* (*ICRQ*). Оно используется для уведомления о том, что между LAC и LNS для этого вызова устанавливается сессия, и передает LNS параметры сессии. В ответ на принятое сообщение *ICRQ* LNS отвечает сообщением *Incoming-Call-Reply* (*ICRP*). Оно позволяет LNS сообщить LAC параметры для данной сессии L2TP. Получив сообщение *ICRP*, в ответ на него LAC отправляет сообщение *Incoming-Call-Connected* (*ICCN*). Оно извещает LNS, что сообщение *ICRP* принято, вызов обслужен, и сессия L2TP должна перейти в стадию «Установлена». LNS может отправить сообщение *ZLB ACK*, если в его очереди больше нет сообщений к LAC.

При установлении исходящего вызова, который инициирует LNS, он отправляет LAC сообщение *Outgoing-Call-Request* (*OCRQ*). Оно используется для уведомления о том, что между LNS и LAC для этого вызова должна быть установлена сессия, и передает LAC параметры сессии L2TP и вызова, который должен быть сделан. LAC отвечает LNS сообщением *Outgoing-Call-Reply* (*OCRP*). Оно сообщает, что LAC пытается выполнить вызов и содержит определенные параметры, относящиеся к этой попытке вызова. После выполнения вызова LAC вслед за сообщением *OCRP* отправляет LNS сообщение *Outgoing-Call-Connected* (*OCCN*). Оно сообщает, что запрошенный исходящий вызов успешно выполнен, и предоставляет

LNS информацию о параметрах, полученных после установления вызова. LNS может отправить сообщение *ZLB ACK*, если в его очереди больше нет сообщений к LAC.

Установление PPP-сессии

После завершения установления L2TP-сессии следующими шагами являются переговоры PPP, назначение удаленной системе внутренних IP-адресов и передача данных PPP (рис. 2.77). В процессе переговоров PPP выполняются протоколы LCP, NCP и необходимые опциональные протоколы, например протоколы аутентификации. После завершения процесса переговоров начинается передача данных PPP.

Кадры PPP передаются как вложенные данные. Они инкапсулируются в пакет L2TP и перенаправляются в соответствующий туннель. LNS получает L2TP-пакет, обрабатывает инкапсулированный кадр PPP, как если бы он был получен на локальном PPP-интерфейсе.

Напомним, что между парой LAC — LNS может существовать множество туннелей, а в каждом туннеле может быть по нескольку L2TP-сессий.

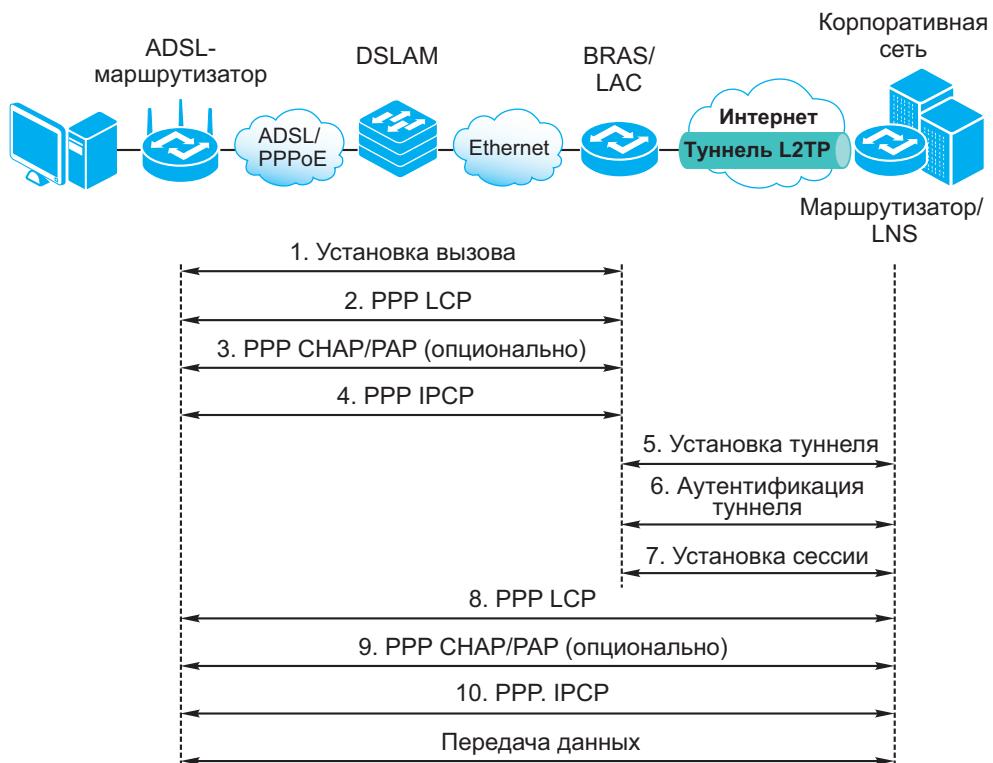


Рис. 2.77. Операции протокола L2TP

2. Протокол PPP

Отправитель сообщения, связанного с конкретной сессией и туннелем, указывает в полях *Session ID* и *Tunnel ID* заголовка пакета L2TP идентификаторы получателя. Таким образом, между LNS и LAC могут передаваться кадры PPP разных туннелей и сессий.

Нулевые значения идентификаторов *Session ID* и *Tunnel ID* используются только при установлении новой сессии.

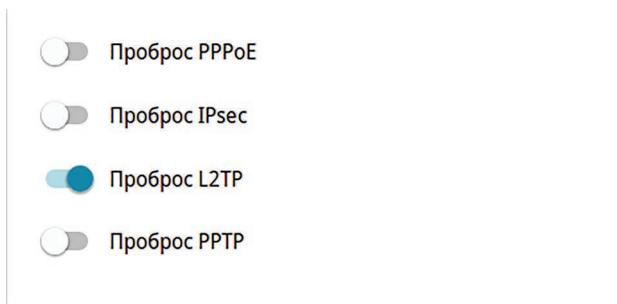


Рис. 2.78. Настройка функции проброса L2TP на маршрутизаторе D-Link

Когда туннель к LNS устанавливает клиент LAC, перед установкой туннеля должно быть создано виртуальное PPP-соединение. Если клиентом LAC является компьютер с установленным ПО L2TP, на абонентском маршрутизаторе, к которому он подключен должна быть активирована функция *L2TP pass through* (проброс L2TP) (рис. 2.78). Эта функция разрешает маршрутизатору пропускать L2TP-трафик, позволяя клиентам L2TP из локальной сети устанавливать соответствующие соединения с LNS.

Завершение сессии, управляющего соединения и туннеля

Завершение сессии, которое может быть инициировано и LAC, и LNS, состоит в отправке управляющего сообщения *Call-Disconnect-Notify* (CDN). После того как последняя сессия завершена, управляющее соединение может быть удалено.

Завершение управляющего соединения может инициироваться и LAC, и LNS. Оно состоит в отправке единственного управляющего сообщения *Stop-Control-Connection-Notification* (StopCCN). Получатель StopCCN отвечает сообщением *ZLB ACK* для подтверждения его получения и поддержания состояния управляющего соединения, если *ZLB ACK* будет потеряно.

Туннель завершается, когда любая из сторон отправляет уведомление *Stop-Control-Connection-Notification*. Перед тем как удалить управляющую информацию, связанную с туннелем, отправитель этого уведомления должен ждать подтверждения в течение определенного периода времени. Получатель данного уведомления должен отправить подтверждение и удалить соответствующую управляющую информацию.

Обеспечение безопасности

L2TP не обеспечивает механизмов безопасности, но позволяет дополнительно выполнять аутентификацию конечных точек при установлении туннеля. Характеристики этой аутентификации такие же, как у CHAP: она защищена от replay-атак и от атак, связанных с подделкой, при установлении туннеля. Однако данный механизм не предполагает никакой аутентификации после того, как туннель установлен. Это приводит к тому, что злоумышленнику достаточно просто вставить вредоносные пакеты в передаваемый PPP-поток после того, как выполнена аутентификация конечных точек туннеля. Для выполнения аутентификации LAC и LNS должны разделять общий секрет. Так как используется единственный секрет, атрибуты AVP, аутентифицирующие туннель, содержат различные значения полей CHAP ID, используемые для вычисления хеш-функции, чтобы гарантировать защиту от replay-атак.

Для обеспечения безопасности L2TP требуется, чтобы нижележащий транспортный протокол мог предоставлять сервисы шифрования, целостности и аутентификации для всего L2TP-трафика. Этот протокол оперирует со всем L2TP-пакетом и функционально не зависит от PPP и протокола, который доставляется по PPP. Таким образом, L2TP предоставляет сервисы конфиденциальности, целостности и аутентификации L2TP-пакетов между конечными точками туннеля (LAC и LNS).

Для обеспечения шифрования, целостности и аутентификации на уровне пакетов L2TP интегрируется с протоколом IPsec (L2TP over IPsec).

IPsec (IP Security) — набор протоколов, предназначенный для безопасной передачи пакетов IP через сеть общего пользования. Он использует протоколы безопасности Authentication Header (AH) и Encapsulating Security Payload (ESP). Все управляющие пакеты и пакеты данных L2TP, относящиеся к конкретному туннелю, являются для IPsec однородными UDP/IP пакетами и инкапсулируются в ESP или AH в зависимости от выбранного алгоритма защиты пакетов.

Следует отметить, что в настройках L2TP-клиентов доступно шифрование MPPE. Это шифрование опционально и используется только для шифрования нагрузки, помещаемой в первоначальный кадр PPP. Сам пакет L2TP не шифруется.

2.13. Типы подключения к провайдерам

В настоящее время существует множество типов подключения к провайдерам (рис. 2.79). Их можно разбить на три основные группы:

1. Подключение по локальной сети IP over Ethernet (IPoE);
2. VPN-подключение на основе протокола PPP;
3. Комбинированное подключение или Dual Access.

Подключение по локальной сети является самым простым способом доступа в Интернет. Пользователь подключается к сети провайдера

2. Протокол PPP

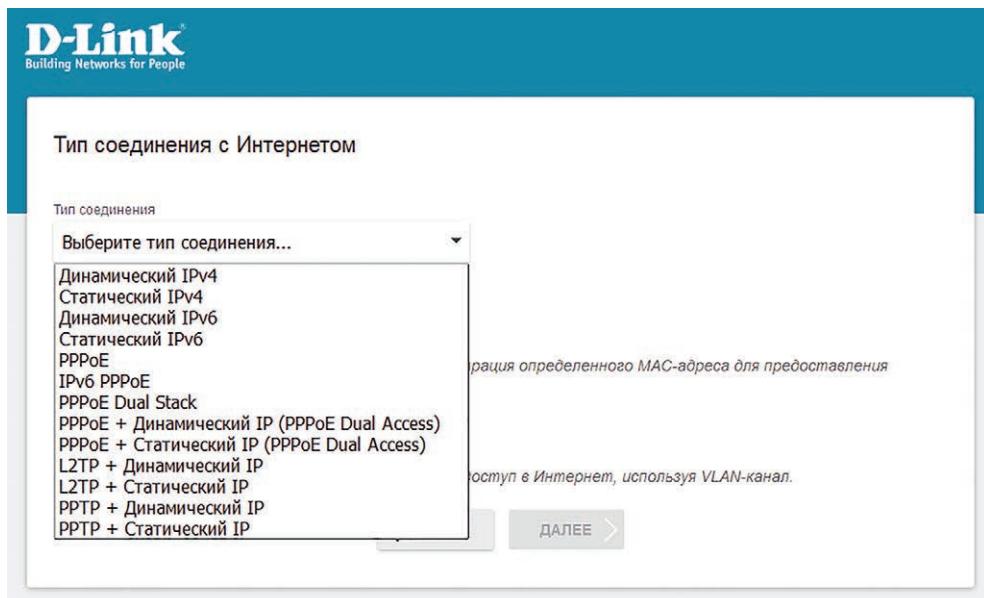


Рис. 2.79. Типы подключения к провайдеру

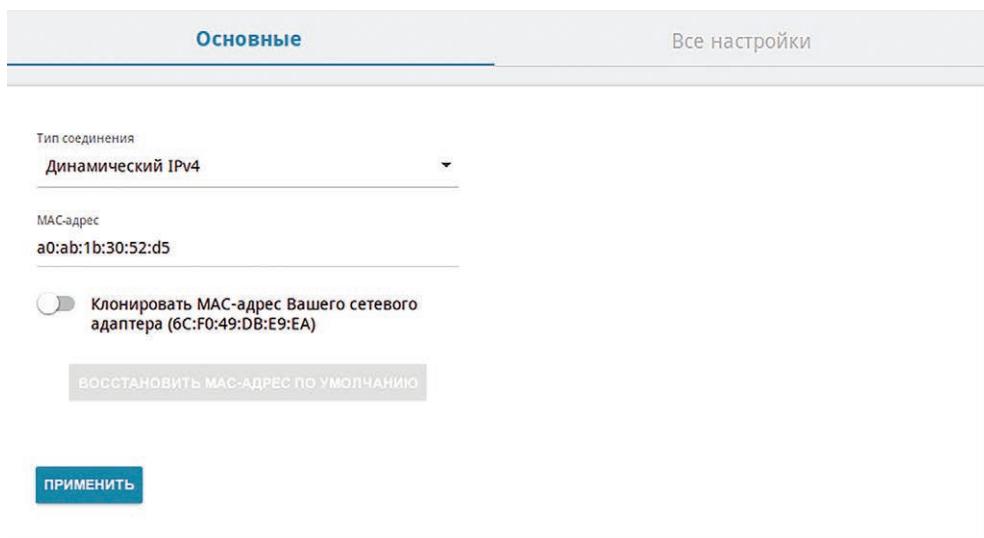


Рис. 2.80. Окно настройки функции «Динамический IP» на маршрутизаторе D-Link

посредством локальной сети Ethernet. Данный тип подключения дополнительно подразделяется на:

- **Dynamic IP/IPv6 (DHCP)** или «Динамический IP/IPv6» (рис. 2.80). Этот тип подключения не требует ввода никаких настроек. Достаточно подключить абонентское устройство кабелем Ethernet к сети провайдера и активировать на нем эту функцию. Все настройки (IP-адрес WAN-интерфейса, IP-адреса шлюза и DNS-серверов) будут получены от провайдера автоматически;
- **Static IP/IPv6** или «Статический IP/IPv6» (рис. 2.81). При конфигурировании абонентского устройства требуется ручной ввод IP-адреса WAN-интерфейса, маски подсети, IP-адресов шлюза и DNS-серверов согласно настройкам, предоставленным провайдером.

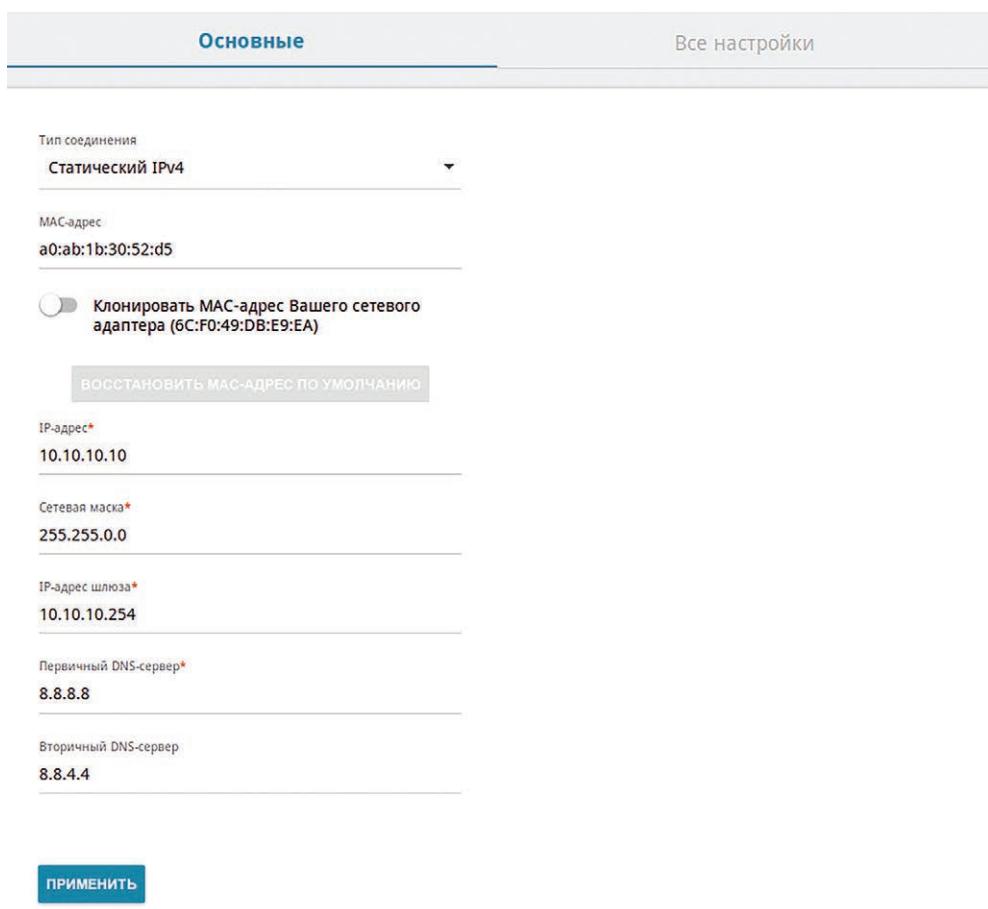


Рис. 2.81. Окно настройки функции «Статический IP» на маршрутизаторе D-Link

2. Протокол PPP

Подключение посредством VPN включает следующие методы доступа, основанные на протоколе PPP: PPPoE, PPTP и L2TP. Данные методы доступа требуют настройки на абонентском устройстве имени пользователя и пароля, предоставленных провайдером.

Третий тип подключения является комбинированным. Он используется только в России, и суть его сводится к комбинированию подключения по локальной сети (статический или динамический IP) с VPN-подключением (PPPoE, PPTP или L2TP) для предоставления двойного доступа (*Dual Access*), т. е. одновременного доступа в Интернет и к внутренним ресурсам провайдера (IPTV, доступ к игровым и файлообменным серверам провайдера и т. д.). Это тип подключения является самым сложным. На WAN-интерфейсе абонентского маршрутизатора создаются два виртуальных соединения (интерфейса). Первое соединение (PPPoE, PPTP или L2TP) используется для авторизации пользователей и предоставления им доступа в Интернет, второе виртуальное соединение — для получения доступа к ресурсам провайдера.

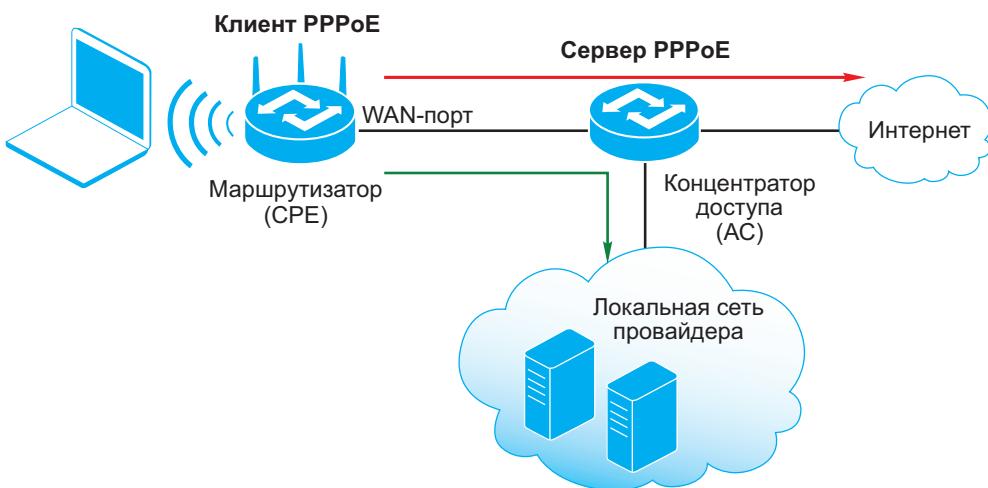


Рис. 2.82. PPPoE Dual Access

В маршрутизаторах D-Link используются следующие комбинации подключений:

- PPPoE + Динамический IP;
- PPPoE + Статический IP;
- PPTP + Динамический IP;
- PPTP + Статический IP;
- L2TP + Динамический IP;
- L2TP + Статический IP.

Настройка функции Dual Access (рис. 2.82) состоит из настройки соединения PPPoE, PPTP или L2TP для доступа в Интернет и подключения

Технологии TCP/IP в современных компьютерных сетях

к локальной сети провайдера. При создании соединения PPPoE, PPTP или L2TP требуется указать имя пользователя и пароль, предоставленные провайдером. Если второе соединения типа «Статический IP», провайдер предоставляет информацию об IP-адресе, маске подсети, IP-адресах шлюза и DNS-серверов, которую требуется ввести в настройках маршрутизатора. Если второе соединения типа «Динамический IP», то все необходимые данные автоматически получаются из сети провайдера.

Внимание: поддерживаемые типы подключения к сети провайдера могут отличаться на разных моделях маршрутизаторов D-Link. Информацию о поддерживаемых конкретной моделью типах подключения можно найти в технических характеристиках устройства на сайте компании www.dlink.ru.

3. Протокол IP

Протокол *IP* (Internet Protocol) является основным протоколом стека TCP/IP и *сетевого уровня* (network layer). Основная задача сетевого уровня — доставка данных между устройствами различных сетей, которые соединены произвольным образом, т. е. образуют *составную сеть* (internetwork) (рис. 3.1).

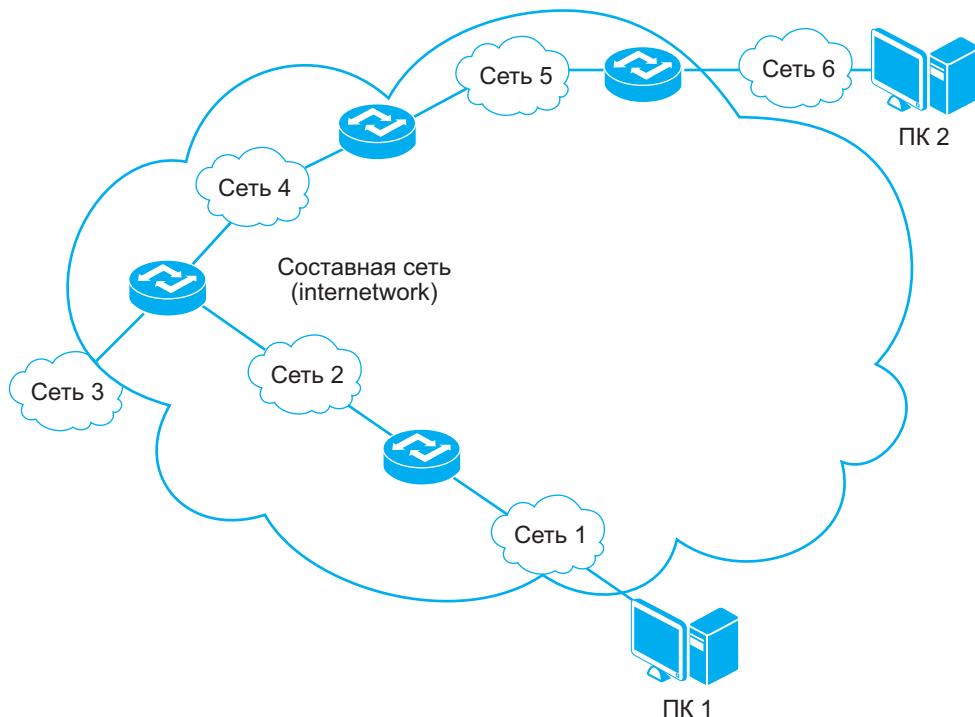


Рис. 3.1. Составная сеть

Сети могут быть построены с использованием различных протоколов канального и физического уровня. Таким образом, они используют различные форматы кадров, методы доступа к среде передачи, методы модуляции и кодирования. Для того чтобы соединить такие сети, нужен общий межсетевой уровень, использующий понятный всем нижележащим сетям протокол. Таким протоколом является протокол IP.

Успех протокола IP был обусловлен его характеристиками, несмотря на имеющиеся в нем ограничения. Он независим от протоколов нижележащих уровней и обеспечивает универсальную адресацию узлов, которая позволяет выполнять маршрутизацию пакетов данных в составных сетях. Протокол IP является протоколом без установления соединения (connectionless protocol). Это означает, что когда узел А хочет передать данные узлу В, им не надо

сначала устанавливать соединение и только после этого получить возможность передачи данных. Протокол IP не использует отправку подтверждений и не гарантирует надежной доставки пакета до адресата. Эта функция выполняется протоколами более высокого уровня. Поэтому его иногда называют *best-effort protocol*. Может показаться удивительным, почему при имеющихся ограничениях (отсутствии установления соединений, ненадежной доставке) протокол получил широкое распространение. Ответ прост — протокол не снижает производительность сети. Установление соединения, надежная доставка и подтверждения требуют времени, ресурсов устройств и сильно снижают пропускную способность сети вследствие передачи большого количества служебных сообщений.

Протокол IP выполняет четыре основные функции:

- адресацию узлов;
- инкапсуляцию данных;
- фрагментацию и последующую сборку пакетов;
- маршрутизацию.

Существует две версии протокола IP: IPv4 (IP version 4) и IPv6 (IP version 6). Как вы уже знаете, первой версией протокола IP стала версия 4, документированная в RFC 760 (январь 1980 года). В сентябре 1981 года RFC 760 был заменен RFC 791. Из-за ограниченного адресного пространства IPv4 появилась необходимость в новой версии протокола IP. Ей стала версия 6 протокола — Internet Protocol version 6 (IPv6), которую также иногда называют IP Next Generation, или IPng. Протокол IPv6 описан в серии RFC начиная с RFC 1883.

Почему новая версия протокола IP не была названа версией 5? Этот номер версии пропустили, чтобы не путать с экспериментальным протоколом стека TCP/IP, получившим название Internet Stream Protocol version 2 (RFC 1190). Предполагалось, что этот протокол станет равноправным IP-протоколом сетевого уровня и его пакетам назначили версию 5, чтобы отличать от оригинальных пакетов IPv4. Протокол Internet Stream Protocol не нашел применения, и чтобы не было путаницы, версию 5 протокола IP пропустили.

3.1. Протокол IP версии 4

Данные, передаваемые с использованием протокола IPv4, помещаются в сообщения, называемые *пакетами* или *дейтаграммами*. Протокол IPv4 использует пакет, который условно можно разделить на заголовок длиной, как правило, 20 байт и на данные. Заголовок содержит адресные и управляющие поля, а в поле *Данные* находится непосредственно информация, которая передается через составную сеть. В отличие от формата некоторых других протоколов, например Ethernet, пакет IPv4 (рис. 3.2) не содержит следующего за полем *Данные* поля контрольной суммы всего пакета.

Пакет IPv4 состоит из следующих полей:

- *Версия* (Version) — для IPv4 значение поля равно 4;

3. Протокол IP



Рис. 3.2. Формат пакета IPv4

- *Длина заголовка* (IHL, Internet Header Length) — указывает на начало блока данных в пакете. Обычно значение для этого поля равно 5;
- *Тип сервиса* (ToS, Type of Service) — содержит информацию, требуемую для обеспечения функций качества обслуживания (QoS);
- *Общая длина* (TL, Total Length) — общая длина пакета с учетом заголовка и поля данных;
- *Идентификатор пакета* (Identification) — используется для распознавания пакетов, образовавшихся путем фрагментации исходного пакета;
- *Флаги* (Flag) — содержит признаки, связанные с фрагментацией пакета (рис. 3.3);

Бит 0: зарезервировано, должен быть 0
Бит 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
Бит 2: (MF) 0 = Last Fragment, 1 = More Fragments.

Биты:	0	1	2
0	D F	M F	

Рис. 3.3. Формат пакета Flag

- *Смещение фрагмента* (Fragment Offset) — значение, определяющее позицию фрагмента в потоке данных;
- *Время жизни* (TTL, Time to Live) — максимальное время в секундах, в течение которого пакет может перемещаться по сети. Маршрутизатор при

получении пакета каждый раз будет уменьшать это время. Как только оно станет равным 0, пакет будет отброшен;

- *Протокол* (Protocol) — указывает, какому протоколу верхнего уровня принадлежит информация, размещенная в поле данных пакета;

- *Контрольная сумма* (Header Checksum) — рассчитывается по заголовку и позволяет определить целостность заголовка пакета;

- *Адрес источника* (Source IP Address) и *адрес назначения* (Destination IP Address) — указывают отправителя и получателя пакета;

- *Опции* (Options) — необязательное поле, может использоваться при отладке работы сети;

- *Данные* (Data) — данные, передаваемые в пакете: или полное сообщение, полученное от вышележащего уровня, или его фрагмент.

Заголовок IPv4, как правило, имеет длину 20 байт. При использовании необязательного поля Опции (Options) длина заголовка может быть увеличена в зависимости от количества опций, но всегда остается кратной 32 битам.

3.1.1. Поле Type of Service

Смысл поля Type of Service (ToS) со временем немного изменился — первоначально (RFC 791) оно предназначалось для обеспечения определенных функций качества обслуживания (QoS) для доставки IP-пакетов (рис. 3.4). Оно позволяло добавлять информацию о приоритете пакета и предпочтительных для его доставки характеристиках сети — задержке, пропускной способности и надежности.

Поле ToS состояло из нескольких подполей. Подполе Precedence длиной 3 бита определяло приоритет пакета. Однобитные поля D (Delay), T (Throughput), R (Reliability) использовались для определения требований к задержке, пропускной способности и надежности сети, по которой передавался IP-пакет. Первые коммерческие маршрутизаторы не поддерживали функции приоритизации. Однако по мере расширения сетей и повышения требований пользователей производители начали добавлять в свои устройства различные виды систем обслуживания очередей, включающие очереди приоритетов, которые обычно реализовывались в виде фильтров, анализирующих поля заголовков протоколов IP (включая Precedence), TCP или UDP.

Подполе Precedence широко использовалось, но не таким образом, как было определено в RFC 791. Биты D, T и R вовсе не применялись в реальных сетях в течение многих лет. В октябре 1989 года появился RFC 1122, в котором было дано новое определение байта ToS. Он был разделен на две части: поле Precedence размерностью 3 бита и поле Type-of-Service, или ToS, длиной 5 бит. В 1998 году появился RFC 2474 «Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers», который переопределил байт ToS для поддержки модели дифференцированного обслуживания (Differentiated Services, DiffServ).

Биты 0-2: Precedence

Бит 3: 0 = Normal Delay, 1 = Low Delay

Бит 4: 0 = Normal Throughput, 1 = High Throughput

Бит 5: 0 = Normal Reliability, 1 = High Reliability

Биты 6-7: зарезервировано

Биты: 0 1 2 3 4 5 6 7

Precedence	D	T	R	0	0
------------	---	---	---	---	---

Precedence

111 – Network Control

110 – Internetwork Control

101 – CRITIC/ECP

100 – Flash Override

011 – Flash

010 – Immediate

001 – Priority

000 – Routine

Рис. 3.4. Формат поля ToS в RFC 791

Напомним, что функции качества обслуживания в современных сетях заключаются в обеспечении гарантированного или дифференцированного уровня обслуживания сетевого трафика, запрашиваемого теми или иными приложениями на основе различных механизмов распределения ресурсов, ограничения интенсивности трафика, обработки очередей и приоритизации.

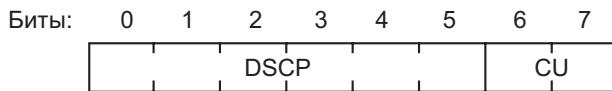
Можно выделить три модели реализации QoS в сети:

- **Негарантированная доставка данных (Best Effort Service)** — обеспечивает связь между узлами, но не гарантирует надежную доставку данных, время доставки, пропускную способность и определенный приоритет;

- **Интегрированные услуги (Integrated Services, IntServ)** — эта модель описана в RFC 1633 и предполагает предварительное резервирование сетевых ресурсов с целью обеспечения предсказуемого поведения сети для приложений, требующих для нормального функционирования гарантированной выделенной полосы пропускания на всем пути следования трафика. Этую модель также часто называют *жестким QoS (hard QoS)* в связи с предъявлением строгих требований к ресурсам сети.

- **Дифференцированное обслуживание (Differentiated Service, DiffServ)** — эта модель описана в RFC 2474, RFC 2475 и предполагает разделение трафика на классы на основе требований к качеству обслуживания. В архитектуре DiffServ каждый передаваемый пакет снабжается информацией, на основании которой принимается решение о его продвижении на каждом промежуточном узле сети, в соответствии с политикой обслуживания трафика данного класса (Per-Hop Behavior, PHB). Модель дифференцированного обслуживания

занимает промежуточное положение между негарантированной доставкой данных и моделью IntServ и сама по себе не предполагает обеспечение гарантий предоставляемых услуг, поэтому дифференцированное обслуживание часто называют *мягким QoS* (*soft QoS*).



DSCP: differentiated services codepoint

CU: не используется

Рис. 3.5. Формат поля DS

Поле ToS в RFC 2474 было заменено на поле DS (Differentiated Services) (рис. 3.5). Первые 6 битов поля DS называются Differentiated Services Codepoint (DSCP) и являются кодовой точкой (codepoint), ассоциированной с классами трафика. Классы определяют политику обслуживания пакета на каждом промежуточном устройстве (коммутаторе, маршрутизаторе), через которое он проходит (Per-Hop Behavior, PHB).

3.1.2. Фрагментация пакетов IPv4

Для того чтобы отправить сообщение, используя протокол IP, данные, полученные от вышележащего уровня, инкапсулируются в IP-пакет. Далее пакет опускается на уровень ниже и инкапсулируется в кадр канального уровня, который помещается в кадр физического уровня и передается по сети. Канальный уровень может использовать разные технологии — Ethernet, 802.11, xDSL, GPON и т. д. Каждая технология канального уровня имеет свои собственные формат кадра и размер поля данных, в которое помещается принятый с сетевого уровня IP-пакет. Например, максимальный размер поля данных стандартного кадра Ethernet составляет 1500 байт, кадра 802.11n — 3839 или 7935 байт. Максимальный размер пакета, который может быть передан через физическую сеть, называется *Maximum Transmission Unit* (MTU).

Обычно узлы стараются отправлять большие пакеты, так как это уменьшает издержки — например, количество служебной информации, такой как заголовки. Когда узел отправляет IP-пакет, он должен определить, не превышает ли размер пакета MTU нижележащего протокола. Если размер пакета больше, то он разбивается на *фрагменты* (fragments), и каждый фрагмент отправляется в виде отдельного пакета сетевого уровня.

Пакет на своем пути от источника до приемника может пройти через множество сетей различных технологий, имеющих разные MTU. Так, при пересечении сети с меньшим значением MTU фрагменты могут быть разделены на более мелкие фрагменты.

Процесс фрагментации пакетов иллюстрирует рис. 3.6. Первоначальный размер IP-пакета равен 12 000 байт. Чтобы передать его через локальную линию связи, компьютер А должен разбить пакет на четыре фрагмента. Маршрутизатор 1 должен передать фрагменты через интерфейс, MTU которого равен 1500 байтам. Таким образом, он разбивает каждый фрагмент на более мелкие фрагменты. Несмотря на то что MTU линии связи между маршрутизаторами 1 и 2 выше, чем между маршрутизаторами 1 и 2, при получении фрагментов размером 1500 байт маршрутизатор 2 не собирает их перед передачей по линии связи с MTU 3839 байт.

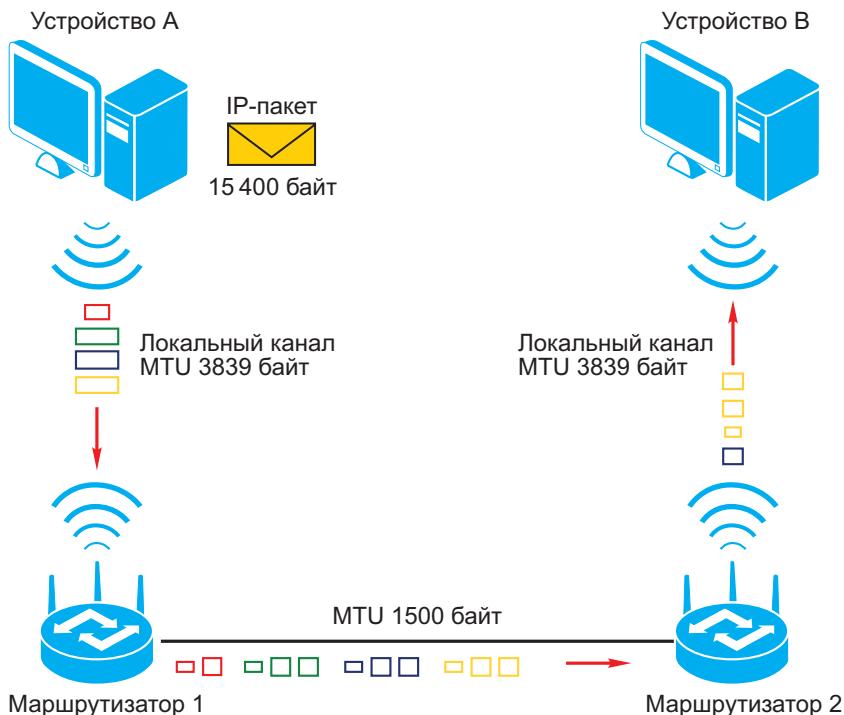


Рис. 3.6. Фрагментация IP-пакетов

Пакет не собирается до тех пор, пока не дойдет до устройства-получателя. Для того чтобы его можно было восстановить, фрагменты пакета должны определенным образом нумероваться. В протоколе IP каждому фрагменту присваиваются идентификатор пакета, абсолютное смещение внутри пакета (в байтах) и флаг, показывающий, является ли этот фрагмент последним в пакете.

При повторной передаче (если не все фрагменты дошли до адресата) пакет может быть разбит на фрагменты по-другому. Размер фрагментов может

быть произвольным, вплоть до одного байта плюс заголовок. В любом случае пакет будет восстановлен: идентификатор пакета и смещение помогут расположить данные в правильном порядке, а флаг укажет на его конец.

Существенным недостатком фрагментации является снижение производительности. Вместо полезной нагрузки передается множество сетевых заголовков, а при потере одного фрагмента требуется повторная передача всего пакета.

Для того чтобы избавиться от фрагментации, надо определить MTU на всем пути следования пакета. Это позволяет выполнить процесс, называемый Path MTU discovery. В своей работе он опирается на сообщения об ошибках протокола *Internet Control Message Protocol (ICMP)*. При отправке каждого пакета IPv4 в его заголовок помещается информация о том, что фрагментация не разрешена (устанавливается флаг Don't Fragment (DF)). Если маршрутизатор получает слишком большой пакет, он отбрасывает его и отправляет источнику сообщение об ошибке Destination Unreachable (рис. 3.7).

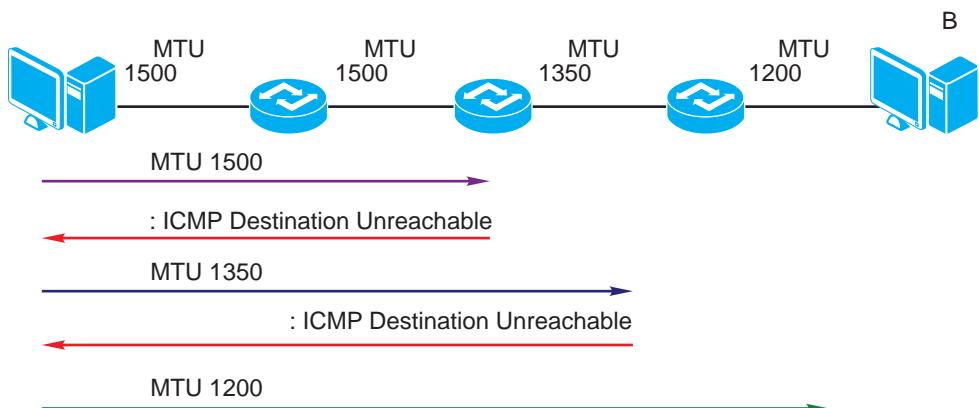


Рис. 3.7. Работа механизма Path MTU discovery

Используя информацию, содержащуюся в сообщении об ошибке, источник уменьшает размер пакетов и отправляет их снова. Если такая же проблема возникнет на каком-нибудь из следующих маршрутизаторов, ситуация повторится.

Преимущество Path MTU discovery состоит в том, что в результате источник знает необходимый размер пакета. При изменении маршрута новое значение MTU станет известно отправителю из новых сообщений об ошибках. Недостатком Path MTU discovery является то, что отправка пакета может вызвать дополнительную задержку. Эта задержка может увеличиться в несколько раз в зависимости от того, сколько раз отправителю придется повторять отправку (меняя размер пакета), прежде чем пакет достигнет места назначения.

3.1.3. Понятие IP-адресации

Основной задачей протокола IP является передача данных между устройствами составной сети, для чего необходима информация о расположении адресата. Идентифицировать адресата и определить маршрут до него позволяет IP-адрес.

В отличие от физического адреса (MAC-адреса), который присваивается каждому сетевому устройству во время изготовления и позволяет уникально идентифицировать каждый узел сети, IP-адрес идентифицирует *сетевой интерфейс* (интерфейс подключения к сети) (рис. 3.8), а не само устройство.

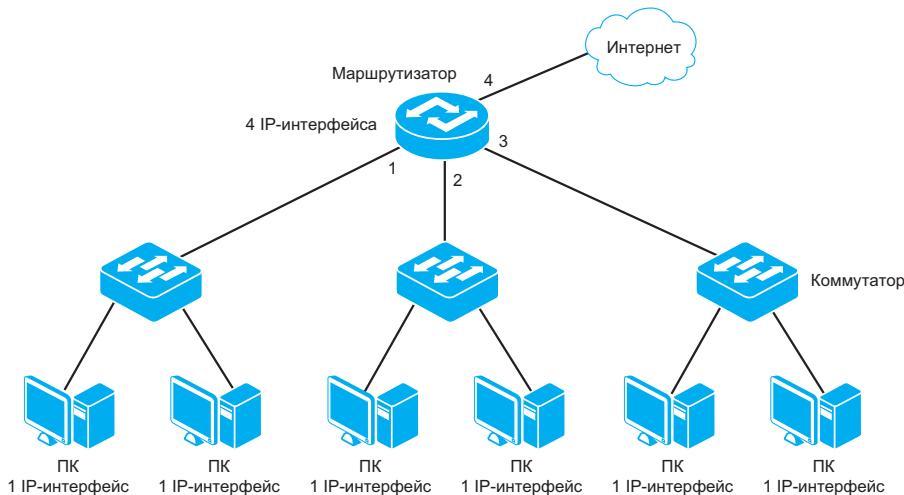


Рис. 3.8. IP-интерфейсы устройств

Любое устройство, которое передает данные, используя сетевой уровень, будет иметь как минимум один уникальный IP-адрес для сетевого интерфейса. Например, таким сетевым узлам, как компьютеры (если установлена одна сетевая карта) и сетевые принт-серверы, обычно присваивают один IP-адрес. Маршрутизаторам или коммутаторам 3-го уровня может быть присвоено более одного IP-адреса, так как они могут использоваться для соединения нескольких сетей.

Для того чтобы устройство участвовало в сетевом взаимодействии с помощью протокола IP, его интерфейсу должен быть присвоен уникальный IP-адрес, который позволяет однозначно идентифицировать интерфейс между ним и данной сетью. IP-адреса назначаются независимо от физических адресов. Если устройство переместить в новую сеть, его IP-адрес изменится, а физический (MAC-адрес) останется прежним.

Таким образом, каждое устройство, которое выполняет передачу данных, имеет связанный с ним адрес на канальном уровне и IP-адрес на сетевом уровне. Возникает вопрос: почему адресация выполняется на двух разных

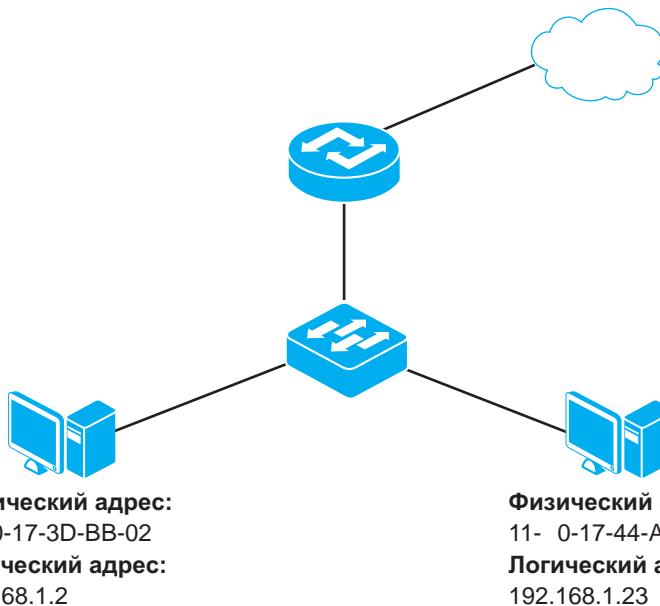


Рис. 3.9. Физические и логические адреса

уровнях? Потому что эти два адреса используются для разных целей. Логически соединение между источником и приемником сообщения в составной сети осуществляется на сетевом уровне с использованием IP-адреса. Поэтому IP-адреса иногда называют логическими адресами (см. рис. 3.9). Физически отправителя и получателя соединяет последовательность каналов связи, работающих по технологиям канального уровня. Чтобы передать данные между непосредственно подключенными устройствами, используются адреса канального уровня, например MAC-адреса.

3.1.4. Представление и структура адреса IPv4

Адрес IPv4 представляет собой 32-разрядное (4 байта) двоичное поле. Для удобства восприятия и запоминания этот адрес разделяют на четыре части по 8 бит (октеты), каждый октет переводят в десятичное число и при записи разделяют точками. Такое представление адреса называется *десятично-точечной нотацией*. Преобразование IP-адреса из двоичного (бинарного) представления в десятичное показано на рис. 3.10.

Следует отметить, что максимальное значение октета равно 11111111 в двоичной системе счисления, что соответствует 255 в десятичной системе счисления, поэтому IP-адреса, в которых хотя бы один октет превышает максимальное значение, считаются недействительными.

Чтобы быстро в уме выполнить преобразование из двоичного вида в десятичный, полезно запомнить табл. 3.1, приведенную ниже. Десятичное

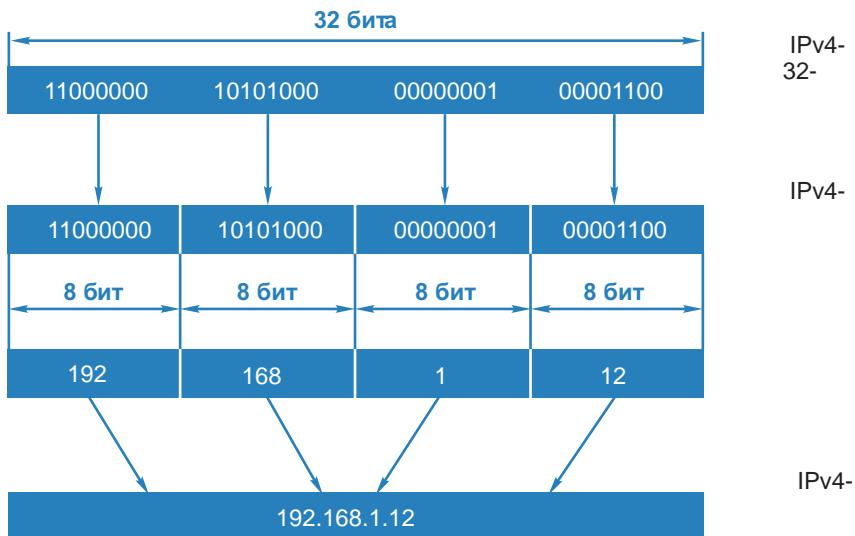


Рис. 3.10. Представление IPv4-адреса в бинарном и десятичном виде

число легко вычисляется как сумма цифр, соответствующих ненулевым битам в октете (см. табл. 3.1).

Таблица 3.1. Преобразование из двоичного вида в десятичный

Двоичное значение октета	Значение битов октета	Десятичное значение октета
00000000	0	0
10000000	128	128
11000000	128+64	192
11100000	128+64+32	224
11110000	128+64+32+16	240
11111000	128+64+32+16+8	248
11111100	128+64+32+16+8+4	252
11111110	128+64+32+16+8+4+2	254
11111111	128+64+32+16+8+4+2+1	255

Маршрутизация пакетов в сетях передачи данных возможна благодаря тому, что IP-адрес структурирован и состоит из двух логических частей: *идентификатора сети* (Net ID) — сетевая часть адреса и *идентификатора узла* (Host ID), который однозначно определяет устройство в сетевом сегменте (рис. 3.11). Такая структура IP-адреса представляет собой двухуровневую иерархическую модель и позволяет устройству при передаче данных в составную сеть указывать не только удаленную сеть, но и узел в этой сети.

**Рис. 3.11.** Структура адреса IPv4

Идентификатор сети определяет конкретную сеть или сегмент сети, в которой находится узел, и используется для передачи данных на нужный сетевой интерфейс маршрутизатора.

После того как данные достигают нужной сети, они передаются уникальному узлу в соответствии с идентификатором узла. Все узлы, использующие один и тот же идентификатор сети, должны быть расположены в одной сети или подсети (логическом сегменте сети).

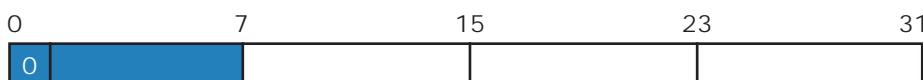
3.1.5. Классовая адресация IPv4

При разработке базовых стандартов и протоколов, положенных в основу будущей Глобальной сети (Интернета), невозможно было представить, какое количество адресов потребуется для работы всех узлов сети. Размер адреса IPv4 был выбран длиной в 32 бита (при этом можно адресовать $2^{32} = 4,3$ млрд устройств). Как показала практика, этой длины адреса для современного Интернета недостаточно. В связи с этим при использовании IPv4 очень важным вопросом является оптимизация выдаваемых адресов с точки зрения максимально эффективного использования адресного пространства IPv4.

Хронологически первым методом разделения IP-адресов является так называемая *классовая модель IP-адресации*, которая частично решила проблему нерационального использования адресного пространства. Согласно этой модели, все пространство IP-адресов делится на пять классов в зависимости от значения первых четырех битов адреса IPv4. Классам присвоены имена от А до Е.

Первые три класса А, В и С используются для индивидуальной (unicast) адресации сетей и узлов, класс D — для многоадресной или групповой (multicast) рассылки, класс Е зарезервирован для экспериментов. Классы А, В и С имеют различную длину сетевой части адреса.

Для сетей класса А под идентификатор сети отводится первый октет, при этом его старший (левый) бит всегда равен 0. Оставшиеся три октета содержат идентификатор узла (рис. 3.12).

**Рис. 3.12.** Формат адреса IPv4 класса А

3. Протокол IP

Поскольку первый бит идентификатора сети всегда равен нулю, то оставшиеся 7 битов позволяют адресовать 128 (2^7) различных сетей. Однако ввиду того, что адреса 0.0.0.0 и 127.0.0.0 являются специальными IPv4-адресами, количество доступных сетей класса А равно 126 ($2^7 - 2$). В каждой сети класса А можно адресовать до 16 777 214 ($2^{24} - 2$) узлов. Два адреса вычтены вследствие того, что они используются в специальных целях и не могут быть назначены устройству (первый — адрес сети, последний — широковещательный адрес).

Сети класса В определяются значениями 10 в двух старших битах адреса (рис. 3.13). Два первых октета адреса содержат идентификатор сети, 3- и 4-й октеты — идентификатор узла. В результате количество доступных сетей класса В составляет 16 384 (2^{14}) с количеством узлов в каждой сети, равным 65 534 ($2^{16} - 2$).

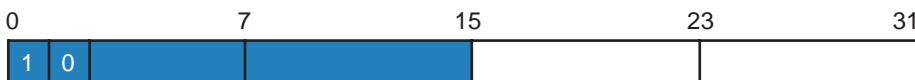


Рис. 3.13. Формат адреса IPv4 класса В

Для сетей класса С под идентификатор сети отводится три октета, а под идентификатор узла — только один октет (рис. 3.14). Три старших бита первого октета всегда равны 110, и это позволяет определить, что адрес относится к классу С. Таким образом, получаем 2 097 152 (2^{21}) сетей, в каждой из которых находится 254 ($2^8 - 2$) узла.

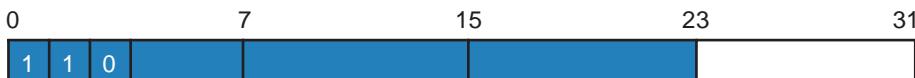


Рис. 3.14. Формат адреса IPv4 класса С

Сети класса D определяются значениями 1110 в первых четырех битах адреса, остальные биты используются для адресации многоадресной группы (рис. 3.15). Адресное пространство класса D зарезервировано для групповой рассылки и используется для адресации группы узлов. Идентификаторов сетей и узлов в IPv4-адресе класса D не выделяют.



Рис. 3.15. Формат адреса IPv4 класса D

Сети класса Е являются экспериментальными и в настоящее время не используются (рис. 3.16). Адреса в этом классе определяются значениями 1111 в первых четырех битах.



Рис. 3.16. Формат адреса IPv4 класса Е

3.1.6. Частные и публичные адреса IPv4

В Интернете идентификация устройств осуществляется уникальными IPv4-адресами, которые не должны повторяться в Глобальной сети. Такие IPv4-адреса называются *публичными* адресами (public addresses). Однако число публичных адресов ограничено, поэтому в каждом из классов IP-сетей определено так называемое *частное пространство IP-адресов* (private addresses). Частные IPv4-адреса предназначены для использования в локальных компьютерных сетях и не маршрутизируются в Интернет. Для локальных сетей, не подключенных к Интернету, можно использовать любые возможные адреса, уникальные в пределах данной сети.

Публичные адреса находятся в пределах от 1.0.0.1 до 223.255.255.254, за исключением частных адресов IPv4.

Адресное пространство частных IPv4-адресов состоит из трех блоков:

- 10.0.0.0 — 10.255.255.255 (класс А);
- 172.16.0.0 — 172.31.255.255 (класс В);
- 192.168.0.0 — 192.168.255.255 (класс С).

Помимо того, определены IPv4-адреса (табл. 3.2), которые имеют специальное назначение (специальные адреса).

Таблица 3.2. Специальные IP-адреса

Идентификатор сети	Идентификатор узла	Описание
Все «0»	Все «0»	0.0.0.0 — адрес узла, сгенерировавшего пакет. Используется устройством для ссылки на самого себя, если оно не знает свой IPv4-адрес. Например, когда устройство пытается получить IPv4-адрес с помощью протокола DHCP
Все «0»	Идентификатор узла	Узел назначения принадлежит той же сети, что и узел-отправитель, например, 0.0.0.25
Идентификатор сети	Все «0»	Адрес IPv4-сети, например, 175.11.0.0
Идентификатор сети	Все «1»	Широковещательный адрес IPv4-сети, например, 192.168.100.255. Узел может отправить широковещательный пакет всем узлам сети/подсети, используя этот адрес

Окончание табл. 3.2

Идентификатор сети	Идентификатор узла	Описание
Все «1»	Все «1»	Ограниченный широковещательный адрес (limited broadcast) 255.255.255.255 никогда не передается за пределы сети/подсети источника. Этот адрес может использоваться, например, узлами, которые не знают идентификатор своей сети и запрашивают его
127.0.0.1		Адрес интерфейса обратной петли (loopback) предназначен для тестирования оборудования без реального отправления пакета

3.1.7. Формирование подсетей

Изначально адрес IPv4 имел два уровня иерархии: идентификатор сети и идентификатор узла. Каждой организации выдавался IPv4-адрес из нужного диапазона (A, B или C) в зависимости от текущего числа компьютеров и его планируемого увеличения.

Для того чтобы более эффективно использовать адресное пространство, были внесены изменения в существующую классовую систему адресации. В RFC 950 была описана процедура разбиения сетей на подсети, и в структуру IPv4-адреса был добавлен еще один уровень иерархии — *подсеть* (subnetwork). Таким образом, была создана трехуровневая иерархия в системе IP-адресации: сеть, содержащая подсеть, каждая из которых включает определенное количество узлов.

Появление еще одного уровня иерархии не изменило самого IPv4-адреса, он остался 32-разрядным, а часть адреса, отведенная ранее под идентификатор узла, была разделена на две части — идентификатор подсети и идентификатор узла (рис. 3.17).



Рис. 3.17. Трехуровневая иерархия IP-адреса

Разбиение одной крупной сети на несколько более мелких подсетей позволяет (рис. 3.18):

- лучше соответствовать физической структуре сети;
- рационально использовать адресное пространство (т. е. для каждого сегмента сети требуется выделять не целиком блок IP-адресов класса A, B или C, а только его часть);

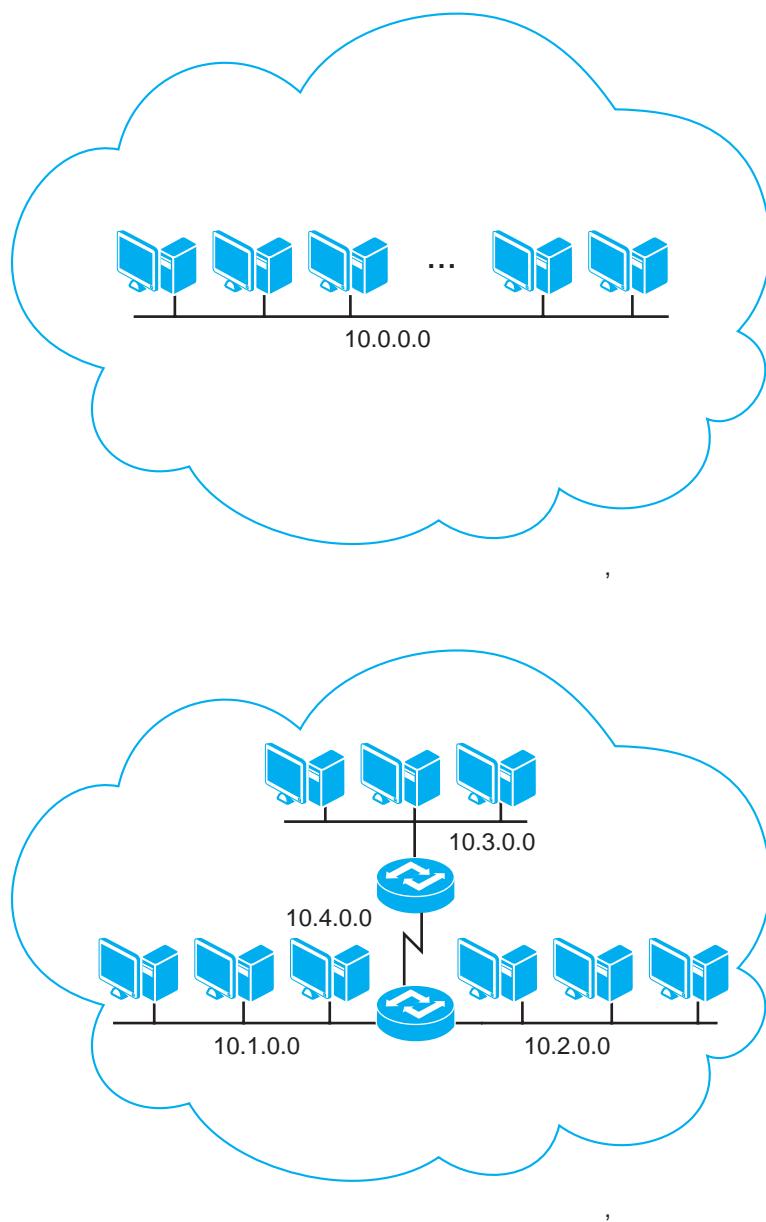


Рис. 3.18. Пример разбиения на подсети

- упростить маршрутизацию;
- повысить безопасность и управляемость сети (благодаря уменьшению размеров сегментов и изоляции трафика сегментов друг от друга).

С появлением трехуровневой иерархии IPv4-адреса потребовались дополнительные методы, которые позволяли бы определить, какая часть адреса указывает на идентификатор подсети, а какая — на идентификатор узла. Было предложено использовать битовую маску (bit mask), которая отделяла бы часть адресного пространства идентификаторов узлов от адресного пространства идентификаторов подсети. Такая битовая маска называется *маской подсети* (subnet mask).

Маска подсети — это 32-битное число, двоичная запись которого содержит непрерывную последовательность единиц в тех разрядах, которые определяют идентификатор подсети и непрерывную последовательность нулей в тех разрядах, которые определяют идентификатор узла (рис. 3.19). Маска записывается в точечно-десятичном представлении аналогично IP-адресу.

IP-	11000000	10101000	00000001	00000010	192.168.1.	2
	11111111	11111111	11111111	00000000	255.255.255.	0
	<hr/>				<hr/>	

Рис. 3.19. Формирование маски подсети

Чтобы получить адрес сети, зная IPv4-адрес и маску подсети, необходимо применить к ним операцию *логическое «И»* (рис. 3.20). Другими словами, в тех позициях IPv4-адреса, в которых в маске подсети стоят двоичные 1, находится идентификатор сети, а где двоичные 0 — идентификатор узла. Во избежание проблем с адресацией и маршрутизацией **все узлы в одном сегменте сети должны использовать одну и ту же маску подсети**.

IP-	11000000	10101000	00000001	00000010	192.168.1.2
	11111111	11111111	11111111	00000000	$\&$ 255.255.255.0
	11000000	10101000	00000001	00000000	$=$ 192.168.1.0

Рис. 3.20. Получение адреса сети из IP-адреса и маски подсети

Для сетей класса A, B и C определены фиксированные маски подсети, которые жестко определяют количество возможных IPv4-адресов и механизм маршрутизации (табл. 3.3).

Таблица 3.3. Маски подсети для стандартных классов сетей

Класс сети	Маска подсети	Количество битов идентификатора сети
A	255.0.0.0	8
B	255.255.0.0	16
C	255.255.255.0	24

При применении масок сети можно разбивать на меньшие по размеру подсети путем расширения сетевой части адреса и уменьшения узловой части. Технология разделения сети дает возможность создавать большее число сетей с меньшим количеством узлов в них, что позволяет эффективно использовать адресное пространство.

Для вычисления количества подсетей используется формула 2^s , где s — количество битов, занятых под идентификатор сети из части, отведенной под идентификатор узла. Количество узлов в каждой подсети вычисляется по формуле $2^n - 2$, где n — количество битов, оставшихся в части,

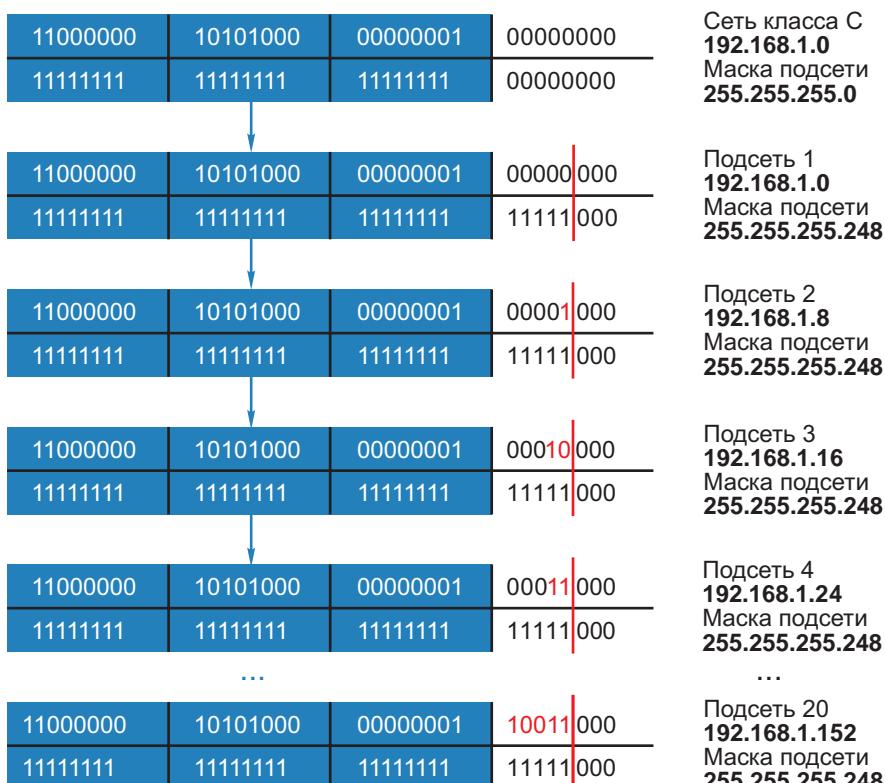


Рис. 3.21. Пример разбиения сети 192.168.1.0 на подсети

идентифицирующей узел, а два адреса — адрес подсети и широковещательный адрес — в каждой полученной подсети зарезервированы.

Рассмотрим пример планирования подсети (рис. 3.21). Какой должна быть маска подсети, если организации необходимо разбить сеть 192.168.1.0 на 20 подсетей по шесть компьютеров в каждой. Для начала необходимо определить, к какому классу относится адрес. Сеть 192.168.1.0 — класс C, соответственно, стандартная маска подсети для класса C равна 255.255.255.0 и под идентификатор узла отведен 4-й октет. Затем определяется количество битов 4-го октета, занимаемых для формирования 20 подсетей. Поскольку невозможно найти число, при котором степень 2 будет равна 20, выбираем ближайшее большее число $2^5 = 32$. Таким образом, 5 первых битов 4-го октета будут использованы для идентификации подсети, а оставшиеся 3 бита — для идентификации узлов в них. Маска подсети должна быть равна 255.255.255.248 (рис. 3.21).

3.1.8. Маски подсети переменной длины (VLSM)

Использование одной маски позволяет организации разбить сеть только на подсети одинакового размера, что приводит к неэффективному использованию адресного пространства, особенно тогда, когда подсети содержат разное количество узлов. Может возникнуть ситуация, при которой в одной из подсетей допустимого количества узлов будет недостаточно, а в другой, наоборот, большая часть адресов не будет использована. Например, большое количество узлов является избыточным для подсети, которая связывает два маршрутизатора по каналу «точка-точка». В этом случае необходимы всего два IPv4-адреса для адресации интерфейсов соседних маршрутизаторов.

Технология *VLSM* (Variable Length Subnet Mask, маска подсети переменной длины) позволяет организации использовать более одной маски подсети внутри того же самого адресного пространства и делить сеть на подсети разных размеров. Она была создана в 1987 году и определена в RFC 1009.

Основная идея VLSM заключается в том, что сеть можно разбить на подсеть, а потом подсеть разбить еще на подсети точно таким же образом, как была разбита первоначальная сеть, т. е. сеть может быть разбита на подсети разных размеров и с разными масками.

Вместо маски подсети в VLSM используется нотация «IP-адрес/длина префикса», аналогичная нотации бесклассовой адресации. Число после «/» означает количество единичных разрядов в маске подсети. Например, сетевой адрес 192.168.1.8 с маской подсети 255.255.255.248 также может быть записан как 192.168.1.8/29. Число 29 указывает, что в маске подсети 255.255.255.248 имеются 29 единичных битов.

Деление сети на подсети с использованием масок переменной длины аналогично традиционному делению на подсети.

Рассмотрим пример, показанный на рис. 3.22. Допустим, организация использует сеть класса C 192.168.1.0/24. Требуется разделить ее на шесть

подсетей. В 1-, 2-, 3- и 4-й подсетях должно быть 10 узлов, в 5-й подсети — 50 узлов, в 6-й подсети — 100.

Теоретически для сети класса С 192.168.1.0/24 допустимое количество узлов равно 254, и разбить такую сеть на подсети с требуемым количеством узлов без использования VLSM невозможно.

Сначала делим сеть 192.168.1.0/24 на две подсети. Для этого из 4-го октета необходимо занять 1 бит для идентификатора подсети, таким образом, для идентификации узлов останется 7 битов. В итоге получается две подсети: 192.168.1.0/25 и 192.168.1.128/25, в каждой из которых может быть по 126 ($2^7 - 2$) узлов. Первую из них оставим, так как требуется, чтобы в 6-й подсети было 100 узлов, а вторую разделим еще на две подсети. Для этого возьмем 1 бит из оставшихся 7 битов, отведенных под идентификатор узла. Таким

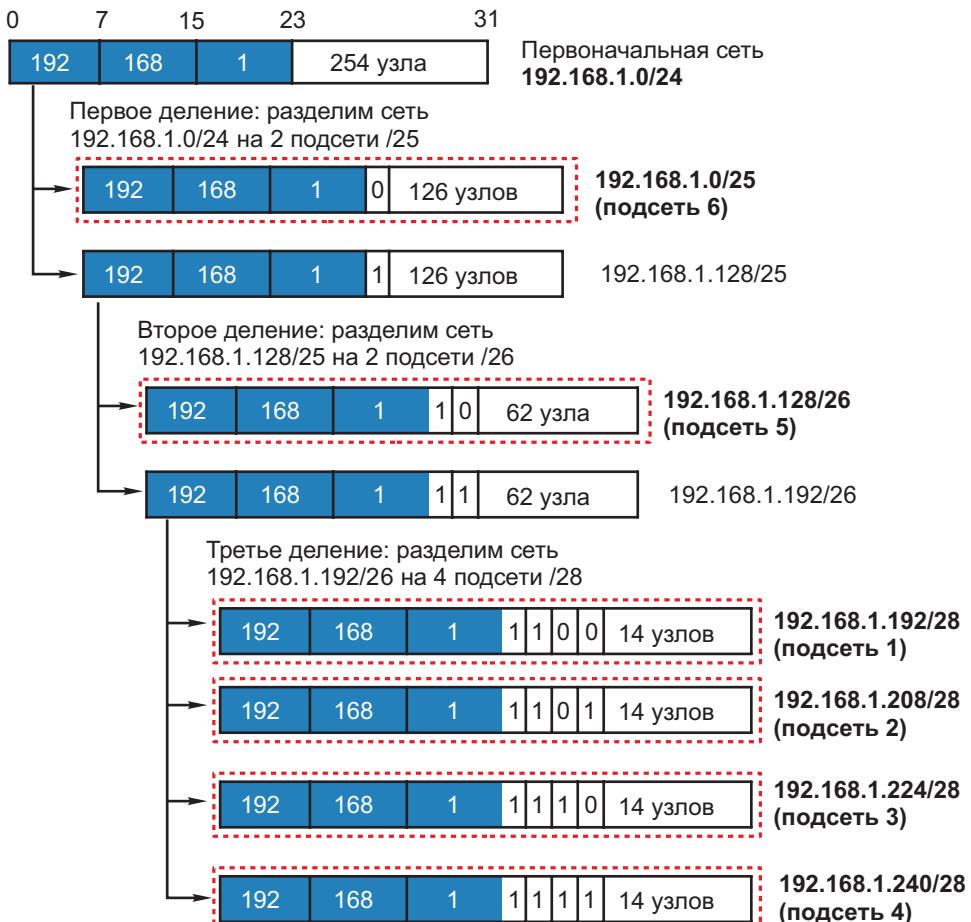


Рис. 3.22. Пример разбиения сети 192.168.1.0/24 на подсети с помощью VLSM

образом, получаются две подсети: 192.168.1.128/26 и 192.168.1.192/26, в каждой из которых допустимое количество узлов равно 62 ($2^6 - 2$). Первую подсеть необходимо оставить для 5-й подсети, в которой должно быть 50 узлов, а из второй подсети сформировать еще четыре подсети. Для этого займем еще 2 бита из оставшихся 6 битов, отведенных под идентификатор узла. В результате получим четыре подсети с 14 ($2^4 - 2$) узлами в каждой, что позволит адресовать требуемое количество узлов, необходимых для подсетей 1, 2, 3 и 4.

VLSM повышает гибкость и эффективность разбиения сетей на подсети. Для использования этой функции в сети необходимо, чтобы маршрутизаторы или коммутаторы L3 поддерживали протоколы маршрутизации, передающие информацию о маске подсети в своих обновлениях.

Технология VLSM похожа на технологию бесклассовой междоменной маршрутизации (Classless Inter Domain Routing, CIDR). Однако между ними существует следующее различие: VLSM имеет дело с подсетями одной сети конкретной организации, а CIDR применяет концепцию разбиения сетей на подсети разных размеров к Интернету в целом.

3.1.9. Бесклассовая адресация IPv4

Классовая модель IPv4-адресации оказалась нерациональной с точки зрения эффективного использования адресного пространства. Например, для сети из 1000 устройств выделялся диапазон адресов класса В, содержащий 65 534 адресов. При этом 1000 адресов использовались, а оставшиеся 64 534 — не использовались.

Разбиение сетей на подсети также не помогло повысить эффективность использования адресного пространства, поскольку оно применялось внутри «классовых» адресных блоков. Кроме того, это не смогло решить проблему экспоненциального увеличения размера таблиц маршрутизации.

Решение проблемы было найдено в отказе от классовой схемы адресации и использовании *бесклассовой* модели. Эта модель была разработана в начале 1990-х годов и формализована в 1993 году в RFC 1517, 1518, 1519 и 1520. Бесклассовая модель адресации получила название *бесклассовой междоменной маршрутизации* (Classless Inter Domain Routing, CIDR).

Несмотря на такое название, CIDR является системой и адресации, и маршрутизации. Она заменила фиксированные классы адресов на гибкую многоуровневую структуру сетей различных размеров и добавила агрегацию маршрутов, известную как *supernetting*.

В классовой схеме адресации IP-адрес имел три уровня иерархии: сеть, подсеть и идентификатор узла. При передаче пакета маршрутизатор определял класс адреса и затем на его основе идентифицировал номер сети и номер узла. В CIDR для определения того, какая часть адреса идентифицирует сеть, а какая — узел, используется битовая маска.

CIDR применяет концепцию VLSM, т. е. деления сети на подсети разных размеров, не к одной конкретной сети, а в целом к Интернету. По сути Интернет становится одной гигантской сетью, которая делится на некоторое

количество больших блоков (больших подсетей). Какие-то из этих больших блоков затем разбиваются на блоки меньших размеров, которые в дальнейшем также могут быть разбиты на еще меньшие блоки. Это разбиение может происходить по нескольку раз, позволяя таким образом разбить адресное пространство Интернета на куски разных размеров, соответствующие требованиям организаций.

Таким образом, бесклассовая адресация полностью исключает понятие классов. Больше не существует блоков адресов класса A, B, C, которые определялись по некоторым первым битам адреса и имели фиксированное количество битов, отведенных под номер сети. При бесклассовой адресации все блоки адресного пространства Интернета имеют произвольный размер.

Для того чтобы провести границу между номером сети и номером узла, CIDR использует маску подсети. Однако CIDR вместо привычной 32-разрядной двоичной маски подсети использует слэш-нотацию (*slash notation*), которую также называют CIDR-нотацией (*CIDR notation*). Это метод записи с помощью косой черты «/». Количество битов, отведенных под идентификатор сети (network ID), которое называется длиной префикса, записывается после «/», следующей за IP-адресом, — «IP-адрес/длина префикса».

Например, запись адреса сети 184.13.152.0/22 говорит о том, что 22 бита в маске подсети отведены под идентификатор сети. Следовательно, для идентификации узлов остается 10 битов. По-другому этот адрес можно записать как IP-адрес 184.13.152.0 с маской подсети 255.255.252.0.

Для использования бесклассовой адресации в сети необходимо, чтобы маршрутизаторы или коммутаторы L3 поддерживали протоколы маршрутизации, передающие информацию о маске подсети в своих обновлениях.

Общие функции классовой и бесклассовой адресации

Существуют несколько аспектов адресации, которые были определены в рамках классовой схемы и перешли без изменения в CIDR:

- блоки частных IP-адресов;
- IP-адреса специального назначения;
- адреса интерфейса обратной петли (loopback).

Выделение адресов

Адресное пространство Интернета выделяется иерархическим образом. Первоначально было два уровня иерархии: IANA (Internet Assigned Numbers Authority, Агентство по выделению имен и уникальных параметров протоколов Интернета) — организация, которая выполняла централизованное выделение блоков адресов, регистрацию доменных имен DNS и публиковала параметры протоколов, например, номера портов TCP и UDP, и различные компании и группы, которым она напрямую выделяла блоки адресов. Управлял IANA Джонатан Постел, один из разработчиков TCP/IP и Интернета.

В конце 1990-х годов появилась новая некоммерческая организация, отвечающая за глобальную координацию системы уникальных элементов

3. Протокол IP

Интернета, ее стабильную работу и безопасную организацию — ICANN (Internet Corporation for Assigned Names and Numbers, Корпорация по управлению доменными именами и IP-адресами). В настоящее время IANA является департаментом ICANN.

Первоначальная схема IP-адресации была основана на классах, поэтому IANA назначала организациям блоки адресов класса A, B и C.

С появлением CIDR IANA перестала выделять адреса непосредственно организациям, и в иерархической структуре выделения адресов появилось больше уровней. IANA делит адресное пространство на большие блоки, которые распределяет среди пяти региональных Интернет-реестров (Regional Internet Registries, RIR): AFRINIC (Африка), APNIC (Азия/Тихоокеанский регион), ARIN (Канада, США и некоторые Карибские острова), LACNIC (Латинская Америка и некоторые Карибские острова) и RIPE NCC (Европа, Ближний Восток и Центральная Азия). Региональные Интернет-реестры далее делят выделенные блоки адресов и выделяют их национальным Интернет-реестрам (National Internet Registries, NIR), локальным Интернет-реестрам (Local Internet Registries, LIR) и/или организациям, таким как провайдеры Интернета.

Провайдеры могут делить полученные блоки адресов на меньшие и выделять их конечным пользователям или менее крупным провайдерским компаниям.

Конечный пользователь, получив адрес, может выполнить деление своей сети на подсети одинаковых или разных размеров с использованием VLSM.

Агрегирование маршрутов и суперсети

Бесклассовая адресация позволяет уменьшить размер таблиц маршрутизации за счет агрегирования маршрутов. При классовой системе адресации, маршрутизаторы должны хранить в таблице маршрутизации записи о маршрутах к каждой сети. Бесклассовая маршрутизация позволяет хранить на маршрутизаторах Интернета только один агрегированный маршрут (supernet route) к сети соответствующего провайдера. Провайдер на маршрутизаторах своей сети хранит записи о маршрутах к сетям своих клиентов.

Далее рассмотрим пример (рис. 3.23). Предположим, что небольшая провайдерская компания получила от крупного провайдера блок адресов 71.94.0.0/15. В нем «/15» означает, что идентификатор сети занимает 15 битов, а остальные 17 битов используются для идентификации узлов. Используя этот блок адресов, можно адресовать 131 070 узлов. Далее провайдер делит этот блок адресов на блоки разных размеров в зависимости от своих потребностей и требований клиентов.

Благодаря CIDR провайдер может иерархически выделять блоки непрерывных IP-адресов, что позволит эффективно использовать адресное пространство и уменьшить количество записей в таблице маршрутизации.

Запишем блок адресов, выделенных провайдеру в двоичном виде:

01000111 01011110 00000000 00000000

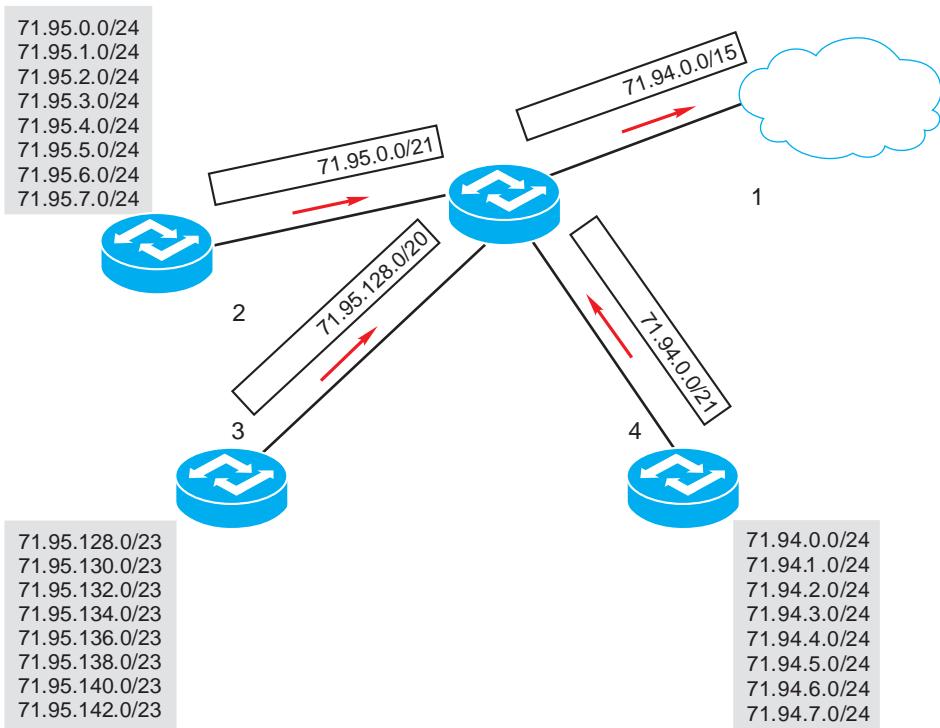


Рис. 3.23. Агрегирование маршрутов

Синим цветом выделены биты, отведенные под идентификатор сети. Красным цветом выделен бит, который будет занят из части, отведенной под идентификатор узла, с целью создания двух подсетей:

Подсеть 0: **01000111 01011110** 00000000 00000000

Подсеть 1: **01000111 01011111** 00000000 00000000

Так как выделенный бит теперь является частью идентификатора сети, получается две подсети с адресами 71.94.0.0/16 (подсеть 0) и 71.95.0.0/16 (подсеть 1). Обратите внимание, что подсеть 0 имеет такой же IP-адрес, как и большая сеть, из которой она появилась. Это справедливо для всех нулевых подсетей.

Пусть адресное пространство подсети 0 провайдер зарезервирует для будущих применений, а подсеть 1 разделит на четыре подсети. Для этого ему надо занять из части, отведенной под идентификатор узла, еще 2 бита.

Подсеть 1: **01000111 01011111 00**000000 00000000

Эти два бита дают четыре комбинации: 00, 01, 10 и 11. Таким образом, для идентификатора сети теперь используются 18 битов. В результате деления подсети 1 получаем четыре подсети с адресами:

3. Протокол IP

Подсеть 1-0: **01000111 01011111 00000000 00000000**
(71.95.0.0/18)

Подсеть 1-1: **01000111 01011111 01000000 00000000**
(71.95.64.0/18)

Подсеть 1-2: **01000111 01011111 10000000 00000000**
(71.95.128.0/18)

Подсеть 1-3: **01000111 01011111 11000000 00000000**
(71.95.192.0/18)

Каждая из подсетей позволяет адресовать по 16 382 узла. Далее каждую из полученных подсетей можно разделить на адресные блоки разных размеров в зависимости от требования потенциальных конечных пользователей. Например, для удовлетворения требований организаций средних размеров, имеющих порядка 250 сотрудников, подсеть 71.95.0.0/18 можно разделить на 64 подсети с длиной префикса 24 бита, т. е. для идентификации сети будут использоваться 24 бита, а для идентификации узлов — 8 битов.

Подсеть 1-0: **01000111 01011111 00000000 00000000**

Адрес первой из 64 подсетей будет 71.95.0.0/24, второй — 71.95.1.0/24, третьей — 71.95.2.0/24, последней — 71.95.63.0/24.

Если провайдер будет выделять клиентам непрерывные блоки IP-адресов, то это позволит агрегировать маршруты к сетям разных клиентов в одну суперсеть (supernetwork). Агрегирование маршрутов, или supernetting, является процессом, обратным разбиению сети на подсети. Агрегирование подразумевает суммирование маршрутов и представление их в таблице маршрутизации в виде одного адреса. Агрегирование выполняется путем выделения в маршрутах общих непрерывных битов высокого порядка.

Например, если провайдер выделил клиентам адреса 71.95.0.0/24 — 71.95.7.0/24, то эти маршруты можно объединить и представить в виде одной записи.

Запишем выделенные блоки адресов в двоичном виде:

01000111 01011111 00000000 00000000 (71.95.0.0/24)

01000111 01011111 00000001 00000000 (71.95.1.0/24)

01000111 01011111 00000010 00000000 (71.95.2.0/24)

01000111 01011111 00000011 00000000 (71.95.3.0/24)

01000111 01011111 00000100 00000000 (71.95.4.0/24)

01000111 01011111 00000101 00000000 (71.95.5.0/24)

01000111 01011111 00000110 00000000 (71.95.6.0/24)

01000111 01011111 00000111 00000000 (71.95.7.0/24)

У всех адресов имеется 21 одинаковый бит высокого порядка:

01000111 01011111 00000

Агрегированный маршрут к этим восьми сетям будет таким:

01000111 01011111 00000000 00000000 (71.95.0.0/21)

Длина префикса равна 21 бит, так как у всех объединяемых маршрутов 21 общий бит.

Агрегирование может выполняться в разных точках сети провайдера до тех пор, пока вся сеть не будет представлена в виде одного агрегированного маршрута 71.94.0.0/15.

Надо отметить, что агрегирование маршрутов возможно только при поддержке маршрутизаторами или коммутаторами L3 протоколов маршрутизации, передающих информацию о маске подсети в своих обновлениях. Агрегирование будет работать правильно, если адреса будут назначаться в иерархическом порядке таким образом, чтобы суммируемые маршруты имели общие биты высокого порядка.

3.1.10. Технология NAT

Бесклассовая адресация помогла улучшить использование адресного пространства IPv4 и на время замедлить его исчерпание. В конце 1990-х годов сокращение адресов IPv4 обещало стать критическим, и инженеры IETF разработали метод, который не только предотвращал исчерпание адресного пространства, но и разрешал две важные проблемы того времени: высокую стоимость IP-адресов и обеспечение безопасности при подключении к Интернету.

Одним из решений этих задач было создание такой системы, где сеть организации *не* подключалась бы *напрямую* в Интернет. Создание такой системы стало возможно благодаря следующим важным особенностям использования Интернета организациями:

- Большинство узлов сети были клиентскими устройствами, и при подключении в Интернет использовалась клиент-серверная модель. Клиентские устройства обычно не делались общедоступными;
- Одновременно в Интернете работали несколько узлов. Когда в организации большое количество узлов подключены в Интернет, в любой момент времени его активно используют небольшое количество клиентов;
- Подключение сети организации в Интернет выполнялось через маршрутизатор, который был точкой контроля трафика.

Можно провести аналогию с телефонной системой организации. Обычно организация имеет один публичный номер телефона, а не получает отдельные номера для каждого сотрудника. Внутри организации используются локальные номера (внутренние номера). Когда кто-то звонит определенному сотруднику организации, он набирает публичный телефонный номер. Далее, если известен внутренний номер сотрудника, можно набрать его или дождаться ответа секретаря, который соединит с нужным человеком.

Для подключения сети организации в Интернет стала использоваться похожая технология. Она получила название The IP Network Address Translator (NAT) и была формализована в мае 1994 года в RFC 1631.

Основные операции NAT заключаются в следующем. Внутри своей локальной сети организация может использовать адреса из диапазона частных адресов. Эти адреса могут использоваться в сетях любых других организаций

сколько угодно раз, поскольку не маршрутизируются в Интернет. Организации также присваивается один или несколько публичных IP-адресов. На границе между локальной сетью организации и публичной сетью устанавливается маршрутизатор с поддержкой функции NAT. Если точек подключения несколько, то NAT-маршрутизаторы устанавливаются в каждой из них. Перед передачей пакетов NAT-маршрутизатор транслирует частные адреса в публичные и наоборот с помощью специальных таблиц трансляции (таблиц NAT). Эти таблицы должны быть одинаковыми на всех маршрутизаторах сети организации. Трансляция заключается в изменении IP-адресов в заголовках пакетов, передаваемых между локальной и публичной сетью.

Статическое и динамическое преобразование адресов

NAT преобразует IPv4-адреса из частного адресного пространства в адресное пространство публичных адресов и наоборот, обеспечивая прозрачную маршрутизацию пакетов между ними. Для трансляции адресов NAT-маршрутизатор использует специальные таблицы. Таблицы трансляции содержат информацию, которая связывает частные IP-адреса устройств локальной сети с глобальными IP-адресами. В некоторых случаях преобразование может быть сделано и для идентификаторов транспортного уровня (таких как порты TCP/UDP). Привязка адресов, если она не существовала, выполняется в начале сессии.

Существует два способа добавления записей в таблицу. Простей формой NAT является статическое преобразование адресного пространства IP. Администратор сети устанавливает статическое однозначное соответствие один к одному между адресами частной сети и адресами внешней сети, которое будет существовать в течение всего времени функционирования NAT.

При динамическом преобразовании соответствие между частными и глобальными адресами автоматически устанавливается NAT-маршрутизатором. Когда сессия, использующая соответствующую привязку адресов, заканчивается, NAT освобождает глобальный адрес, чтобы в дальнейшем он мог опять использоваться для другого узла из частной сети. Конкретный способ связывания адресов специфичен для каждой реализации NAT.

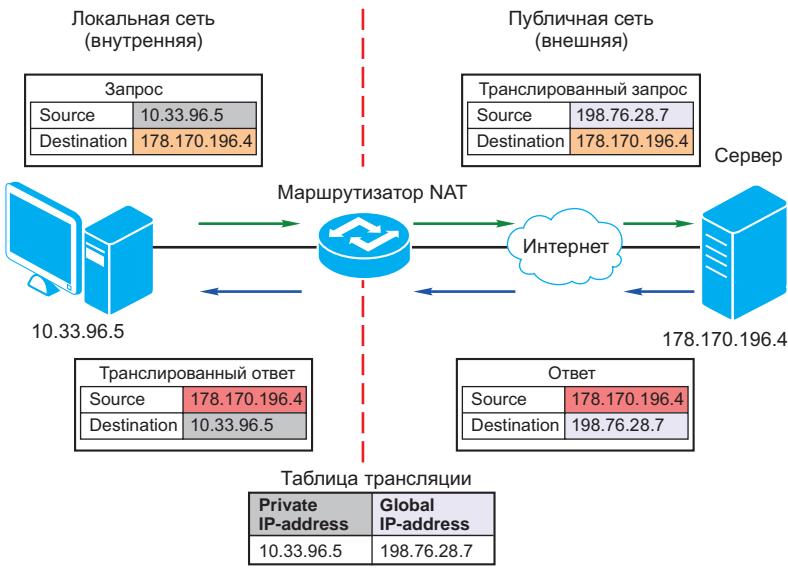
Варианты выполнения NAT

Существует много вариантов выполнения преобразования адресов.

Традиционный (или *исходящий*) NAT позволяет узлам частной сети прозрачно получать доступ к узлам во внешней сети. В традиционном NAT сессия может существовать только в одном направлении, исходящем из частной сети.

Имеются два варианта традиционного NAT, называемых базовым NAT и NAPT (Network Address Port Translation) (рис. 3.24). При базовом NAT в исходящих из локальной сети пакетах NAT-маршрутизатор заменяет частный IP-адрес источника на глобальный IP-адрес из пула адресов и вносит запись в таблицу NAT, где фиксируется соответствие IP-адресов. Затем он рассчитывает новую контрольную сумму для заголовка IP, и измененный

NAT



NATP

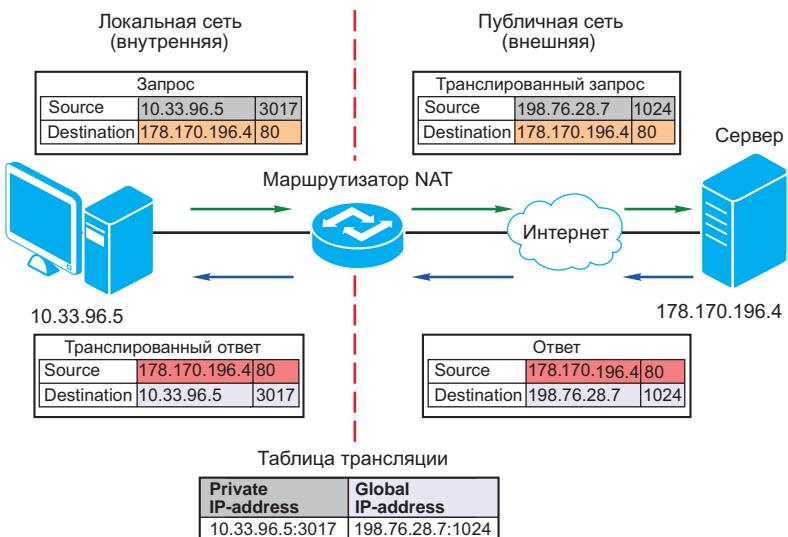


Рис. 3.24. Традиционный NAT:
а — базовый NAT; б — NATP

пакет IP с новым заголовком передается адресату. Адрес получателя в пакете не изменяется.

При получении пакета NAT-маршрутизатор сравнивает IP-адрес назначения с записями в таблице NAT, соответствующим образом изменяет глобальный адрес на частный, после чего создает новую контрольную сумму для заголовка IP и передает измененный пакет получателю во внутренней сети.

NAPT (или Port Address Translation, PAT) дополнительно преобразует идентификатор транспорта, такой как номера портов TCP и UDP. Кроме того, NAPT позволяет большому числу узлов частной сети разделять единственный глобальный адрес. Чтобы можно было различать IP-пакеты различных отправителей, устройство NAPT заменяет номер (возможно, неоднозначный) порта TCP/UDP в заголовке TCP/UDP исходного пакета IP на другой, уникальный номер порта TCP/UDP и вносит соответствующую запись в таблицу NAT. При таком преобразовании система должна заново рассчитать контрольную сумму не только заголовка IP, но и заголовка TCP/UDP. После этого она создает новые заголовки TCP/UDP и IP и передает пакет IP соответствующему адресату.

Заметим, что NAPT может быть скомбинирован с базовым NAT таким образом, чтобы использовался пул глобальных адресов совместно с преобразованием портов.

Двунаправленный NAT обычно используется в том случае, если внешнему устройству надо обеспечить доступ к локальному серверу. При двунаправленном NAT сессии могут инициализировать как узлы из внешней сети (входящее преобразование), так и узлы из частной сети (исходящее преобразование). Выполнение входящих преобразований сложнее, чем исходящих. Устройства частной сети обычно знают IP-адреса внешних устройств. Например, устройство внутренней сети обращается к хорошо известному поисковому серверу. Внешние устройства при этом не знают частных IP-адресов внутри сети. Даже если бы они их знали, то все равно не могли бы их указать в качестве адресов назначения, поскольку частные IP-адреса не маршрутизируются в Интернет. Прозрачная маршрутизация для сессий, начинающихся с узлов из внешней сети, может быть реализована несколькими способами.

1. NAT должен быть сконфигурирован таким образом, чтобы отображать определенный порт получателя на конкретный узел и порт за NAT.

2. Узлу в частной сети может быть назначен IP-адрес, доступный из внешней сети.

Достоинства и недостатки NAT

Технология NAT является одной из тех технологий, которые имеют примерно одинаковое количество достоинств и недостатков.

К достоинствам этой технологии можно отнести следующие:

- использование большим количеством узлов частных IP-адресов;
- простота расширения локальных сетей организаций;

• больший локальный контроль, так как администратор получает возможность контролировать пакеты, передающиеся внутри локальной сети;

- большая гибкость при изменении публичных адресов организаций. Провайдерам стало проще изменять публичные адреса, выделенные клиентам, поскольку количество этих адресов уменьшилось;
- повышение защищенности, так как между локальной и публичной сетями создается межсетевой экран, не позволяющий злонамеренным пользователям подключаться к ресурсам локальной сети;
- NAT выполняется прозрачно для клиентов, т. е. не требует от них выполнения никаких дополнительных действий. Все операции выполняет маршрутизатор.

Недостатки технологии NAT следующие:

- сложность в настройке и управлении;
- проблемы совместимости с определенными приложениями, содержащими внутри себя информацию об IP-адресах;
- проблемы с протоколами шифрования, такими как IPSec;
- уменьшение производительности, поскольку на изменение пакетов требуется время.

3.1.11. Многоадресная передача пакетов IPv4

В современных IP-сетях существуют три способа отправки пакетов от источника к приемнику:

- одноадресная передача (*Unicast*);
- широковещательная передача (*Broadcast*);
- многоадресная передача (*Multicast*).

При *одноадресной передаче* поток данных передается от узла-отправителя на индивидуальный IP-адрес конкретного узла-получателя. *Широковещательная передача* предусматривает доставку потока данных от узла-отправителя на множество узлов-получателей, подключенных к сети, используя широковещательный IP-адрес.

Многоадресная передача обеспечивает доставку потока данных группе узлов на IP-адрес группы *многоадресной рассылки* (групповой IP-адрес).

Многоадресная передача существовала с того момента, как протокол IPv4 был впервые официально определен, но долгие годы не получала широкого применения из-за отсутствия поддержки этой функции в устройствах. Со временем появился интерес к многоадресной передаче, и ее поддержка стала частью стандарта IPv6.

Многоадресная передача имеет ряд преимуществ при работе таких приложений, как видеоконференции, корпоративная связь, дистанционное обучение, видео- и аудиотрансляции и т. д., так как позволяет значительно повысить эффективность использования полосы пропускания и распределения информации среди больших групп получателей. Во-первых, отправитель может один раз передать единственную копию пакета данных всем членам группы, а не рассыпать множество его копий. Во-вторых, благодаря передаче только одной копии пакета снижается нагрузка на канал связи.

Многоадресная передача сложнее одноадресной или широковещательной. Она должна выполнять три основные функции: адресацию, управление группами многоадресной рассылки и обработку/маршрутизацию пакетов многоадресной рассылки.

Адресация многоадресной передачи

Для многоадресной передачи используются специальные IP-адреса. Источник многоадресного трафика направляет пакеты многоадресной рассылки не на индивидуальные IP-адреса каждого из узлов-получателей, а на групповой IP-адрес. Групповые адреса определяют произвольную группу IP-узлов, желающих получать адресованный ей трафик.

Агентство IANA, которое управляет назначением групповых адресов, определило для многоадресной рассылки адреса IPv4 класса D в диапазоне от 224.0.0.0 до 239.255.255.255. Адреса, назначенные IANA, приведены в табл. 3.4. Более подробную информацию о зарегистрированных адресах можно получить на Web-сайте: <http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml#multicast-addresses-12>

Таблица 3.4. Назначенные IANA диапазоны адресов многоадресной рассылки IPv4

Диапазон	Описание
224.0.0.0–224.0.0.255	Блок управления локальной сети (Local Network Control Block). Адреса этого диапазона зарезервированы для использования сетевыми протоколами в сегментах локальных сетей
224.0.1.0–224.0.1.255	Межсетевой блок управления (Internetwork Control Block). Адреса из этого диапазона используются для трафика управления протоколов, который может быть передан через Интернет
224.0.2.0–224.0.255.255	Блок AD-HOC I (AD-HOC Block I). Используется для приложений, которые не попадают в блок управления локальной сетью и межсетевой блок управления
224.1.0.0–224.1.255.255	Зарезервировано
224.2.0.0–224.2.255.255	Блок SDP/SAP (SDP/SAP Block). Этот диапазон адресов предназначен для приложений, которые получают адреса через протокол SAP для использования через приложения, подобные SDR
224.3.0.0–224.4.255.255	Блок AD-HOC II (AD-HOC Block II). Используется для приложений, которые не попадают в блок управления локальной сетью и межсетевой блок управления.
224.5.0.0–224.255.255.255	Зарезервировано
225.0.0.0–231.255.255.255	Зарезервировано

Диапазон	Описание
232.0.0.0–232.255.255.255	Блок специфичной для источника многоадресной рассылки (Source-Specific Multicast Block). Этот диапазон адресов зарезервирован для протокола SSM, который представляет собой расширение протокола PIM
233.0.0.0– 233.251.255.255	Блок GLOP (GLOP Block). Этот диапазон адресов зарезервирован для использования в качестве адресов, статически определяемых организациями с зарезервированным номером автономной системы
233.252.0.0233.255.255.255	Блок AD-HOC III (AD-HOC Block III). Этот диапазон известен как расширенный GLOP (EGLOP, Extended GLOP)
234.0.0.0238.255.255.255	Зарезервировано
239.0.0.0239.255.255.255	Блок административно ограниченных адресов (Administratively Scoped Block). Эти адреса могут локально использоваться внутри домена

Использование групповых адресов IPv4 из блока с административным ограничением наиболее удобно при организации многоадресной рассылки в локальной сети предприятия или организации. В соответствии с RFC 2365 «Administratively Scoped IP Multicast» подсеть 239.192.0.0/14 выделена для частного использования и определена как локальная область организации (IPv4 Organization Local Scope).

Управление группами многоадресной рассылки

Управление группой многоадресной рассылки в сетях IPv4 выполняется с помощью протокола *IGMP* (Internet Group Management Protocol, межсетевой протокол управления группами). Он используется для динамической регистрации отдельных узлов в многоадресной группе и определяет форматы сообщений, которые позволяют передавать информацию о группе и членах группы между устройствами и маршрутизаторами.

В настоящее время существуют три версии протокола IGMP:

- IGMP версии 1 (IGMP v1, описан в RFC 1112);
- IGMP версии 2 (IGMP v2, описан в RFC 2236);
- IGMP версии 3 (IGMP v3, описан в RFC 3376).

Протокол IGMP используется только в сетях с адресацией IPv4. В сетях с адресацией IPv6 групповая передача пакетов реализована на основе протокола MLD (Multicast Listener Discovery).

Обработка и маршрутизация пакетов многоадресной рассылки

Обработка и маршрутизация пакетов многоадресной рассылки является сложной задачей. Так как одно устройство отправляет пакет некоторому множеству устройств, должно быть создано несколько копий этого пакета.

Маршрутизаторы должны понимать, когда они должны создавать эти копии. Также маршрутизаторы должны использовать специальные алгоритмы, которые позволяют определять маршруты отправки многоадресных пакетов. Поскольку создается несколько копий одного пакета, отправляемых в разные места, важно избежать создания избыточного объема трафика. Маршрутизаторы также должны уметь оправлять пакеты членам группы многоадресной рассылки даже в том случае, если источник не является ее членом.

Маршрутизаторы, используемые для организации многоадресной рассылки, должны обладать расширенной функциональностью и включать поддержку протоколов маршрутизации пакетов многоадресной рассылки, таких как *DVMRP* (Distance Vector Multicast Routing Protocol, Протокол дистанционно-векторной многоадресной маршрутизации) или *PIM* (Protocol Independent Multicasts, Многоадресная рассылка, независимая от протокола). Данные протоколы позволяют доставлять пакеты членам многоадресной группы независимо от их местонахождения, избегая при этом передачи избыточного трафика через составную сеть.

3.2. Протокол IP версии 6

Основную проблему протокола IPv4 — относительно небольшое адресное пространство, связанное с использованием адреса длиной 32 бита и его быстрое исчерпание, пытались решить путем перехода на бесклассовую модель адресации, а также с помощью технологии NAT. Однако эти решения только замедлили на какое-то время стремительное исчерпание адресов IPv4. Стало понятно, что 32-разрядное адресное пространство слишком мало для увеличивающегося в размерах Интернета. Это явилось основным фактором появления новой версии протокола IP — протокола IP версии 6 (IPv6). Основные операции IPv6 описаны в RFC 2460 — 2467, опубликованных в 1998 году. Наиболее значимые из них — это RFC 2460 (*Internet Protocol, Version 6 (IPv6) Specification*) и два документа, описывающие протоколы-помощники IPv6: RFC 2461 (*Neighbor Discovery for IP Version 6 (IPv6)*) и RFC 2463 (*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*). Адресацию IPv6 описывали опубликованные в 1998 году RFC 2373 (*IP Version 6 Addressing Architecture*) и RFC 2374 (*An IPv6 Aggregatable Global Unicast Address Format*). В 2003 году эти документы были обновлены, и появились RFC 3513 (*Internet Protocol Version 6 (IPv6) Addressing Architecture*) и RFC 3587 (*IPv6 Global Unicast Address Format*). В 2006 году RFC 3513 был заменен RFC 4291.

Основные отличия IPv6 от IPv4 заключаются не только в увеличении адресного пространства. При разработке IPv6 ставились и другие не менее важные цели. Приведем основные отличия протокола IPv6 от IPv4:

- *Большее адресное пространство.* Размер адреса IPv6 увеличен до 128 битов. Это позволяет адресовать примерно $3,4 \times 10^{38}$ интерфейсов устройств;
- *Иерархическое назначение индивидуальных адресов.* Был создан формат индивидуальных адресов Global Unicast, который используется для

идентификации устройств в Глобальной сети. С его помощью можно создавать множество уровней в иерархической структуре адресов как провайдеров, так и организаций (т. е. множество раз делить большие блоки адресов на меньшие);

- *Расширена поддержка групповых адресов.* Улучшена поддержка групповых адресов и добавлена поддержка нового типа адресации: *альтернативная* (anycast) адресация. Новый тип адресации позволяет доставлять сообщение ближайшему интерфейсу, входящему в группу интерфейсов, идентифицируемых одним адресом;

- *Автоконфигурация.* В протоколе IPv6 узел может практически самостоятельно сконфигурировать параметры своих интерфейсов;

- *Новый формат дейтаграммы.* Формат IP-дейтаграммы был переопределен. К основному (фиксированному) заголовку каждой дейтаграммы IPv6 может быть добавлено произвольное число расширенных заголовков, содержащих требуемую контрольную информацию;

- *Поддержка качества обслуживания (QoS).* Дейтаграмма IPv6 включает функции QoS, обеспечивающие поддержку трафика приложений, чувствительных к задержкам;

- *Поддержка безопасности.* Функции безопасности поддерживаются с помощью расширенных заголовков Authentication и Encapsulation Security Payload.

В связи с множеством различий протоколов IPv4 и IPv6 и важностью протокола IP для функционирования глобальной сети был разработан план по переходу с протокола IPv4 на IPv6. IETF работала над методами, которые позволили бы плавно перейти с протокола версии 4 на протокол версии 6 и обеспечить возможность узлам IPv4 и IPv6 получать доступ друг к другу (рис. 3.25). Эти методы включают:

- **Использование устройств «Dual Stack» (двойной стек):** программное обеспечение устройств включает поддержку обеих версий IP (IPv4 и IPv6), что позволяет им взаимодействовать как с устройствами IPv4, так и с устройствами IPv6;

- **Трансляцию IPv4/IPv6:** программное обеспечение устройств «Dual Stack» может поддерживать функционал, который принимает запросы от узлов IPv6, преобразует их в дейтаграммы IPv4 и отправляет узлам-получателям с поддержкой IPv4. Дейтаграммы от узлов IPv4, предназначенные узлам IPv6, аналогичным образом преобразуются на устройствах «Dual Stack».

- **Туннелирование IPv6 поверх IPv4:** устройства IPv6, на пути между которыми отсутствуют маршрутизаторы IPv6, могут взаимодействовать друг с другом посредством инкапсуляции дейтаграмм IPv6 в дейтаграммы IPv4. Другими словами, они будут использовать два сетевых уровня: IPv6 будет находиться над IPv4. Инкапсулированная дейтаграмма IPv4 будет передаваться между маршрутизаторами IPv4 до тех пор, пока не достигнет адресата.

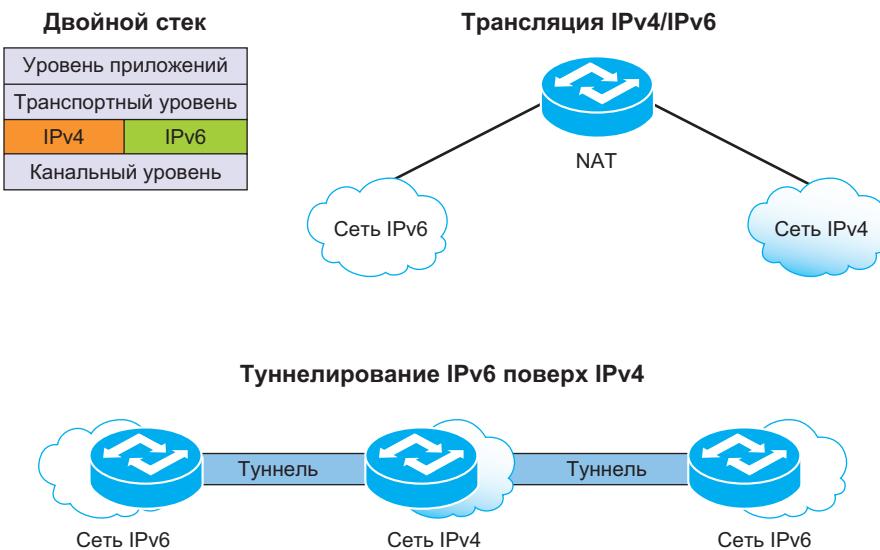


Рис. 3.25. Методы перехода с IPv4 на IPv6

3.2.1. Формат заголовка IPv6

При разработке протокола IPv6 были внесены изменения в формат IP-пакета. Увеличение размера IP-адреса с 32 бит до 128 бит добавило 24 байта к заголовку пакета, что в свою очередь привело к попытке уменьшить его размер вследствие исключения полей, связанных с фрагментацией, и поля контрольной суммы. В результате заголовок пакета IPv6 увеличился всего в 2 раза (с 20 до 40 байт).

Пакет протокола IPv6 состоит из следующего:

- фиксированного заголовка длиной 40 байт, содержащего адреса источника и приемника и всю требуемую для доставки любой дейтаграммы информацию;
- произвольного числа расширенных заголовков (extension header) переменной длины, каждый из которых содержит дополнительную информацию для поддержки таких функций, как фрагментация, маршрутизация, аутентификация, безопасность и опции;
- поля данных переменной длины.

Такой порядок способствует эффективной обработке пакетов на всем пути их следования.

Формат фиксированного заголовка IPv6 показан на рис. 3.26. Он состоит из следующих полей:

- *Версия (Version)* — для IPv6 значение поля должно быть равно 6;
- *Класс трафика (Traffic Class)* — поле используется для идентификации разных классов трафика или приоритетов для обеспечения дифференцированного обслуживания пакетов IPv6;

Технологии TCP/IP в современных компьютерных сетях

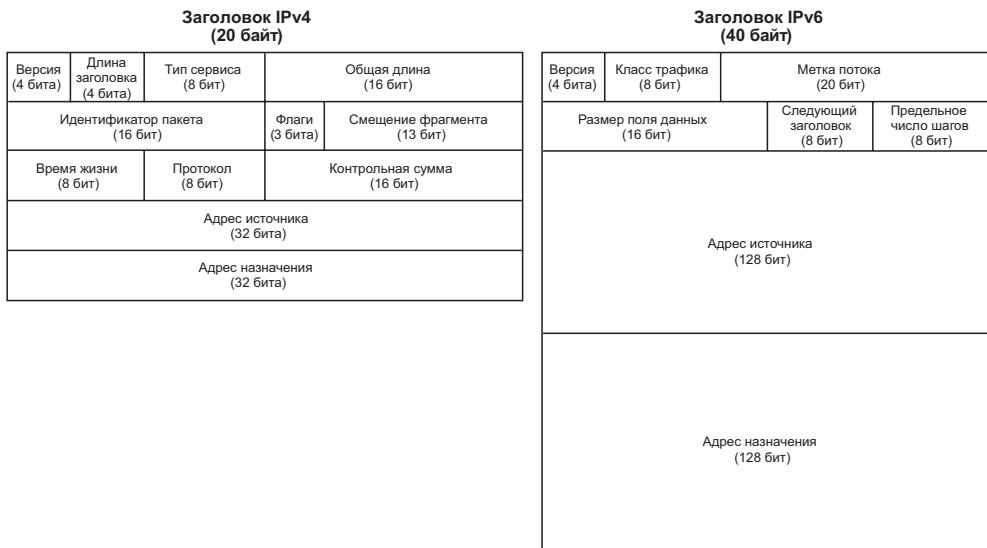


Рис. 3.26. Сравнение форматов заголовка IPv4 и IPv6

- *Метка потока* (Flow Label) — используется отправителем для обозначения последовательности пакетов, которые должны быть подвергнуты определенной обработке маршрутизаторами;
- *Размер поля данных* (Payload Length) — число, указывающее длину поля данных, идущего за заголовком пакета (с учетом расширенного заголовка);
- *Следующий заголовок* (Next Header) — задает тип расширенного заголовка IPv6, который следует за фиксированным;
- *Предельное число шагов* (Hop Limit) — уменьшается на 1 каждым маршрутизатором, через который передается пакет; при значении, равном 0, пакет отбрасывается;
- *Адрес источника* (Source Address) — 128-битный адрес отправителя пакета;
- *Адрес назначения* (Destination Address) — 128-битный адрес получателя пакета.

```

Internet Protocol Version 6, Src: fe80::f1a8:117d:6b9:74a4 (fe80::f1a8:117d:6b9:74a4), Dst: fe80::1 (fe80::1)
⊕ 0110 . . . = Version: 6
⊕ . . . 0000 0000 . . . . . = Traffic class: 0x00000000
    . . . 0000 00. . . . . = Differentiated Services Field: Default (0x00000000)
    . . . . .0. . . . . = ECN-Capable Transport (ECT): Not set
    . . . . .0. . . . . = ECN-CE: Not set
    . . . . .0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 32
Next header: ICMPv6 (0x3a)
Hop limit: 255
Source: fe80::f1a8:117d:6b9:74a4 (fe80::f1a8:117d:6b9:74a4)
Destination: fe80::1 (fe80::1)

```

Рис. 3.27. Заголовок пакета IPv6

Расширенные заголовки IPv6 используются для поддержки механизмов маршрутизации, безопасности, фрагментации, сетевого управления и расположены между фиксированным заголовком и заголовком протокола более высокого уровня. Пакет IPv6 может содержать 0, 1 или несколько расширенных заголовков, каждый из которых определяется значением поля Next Header предшествующего заголовка. Все существующие типы расширенных заголовков описаны в табл. 3.5.

Таблица 3.5. Типы расширенных заголовков IPv6

Расширенный заголовок	Тип	Описание
Hop-by-Hop Options	0	Параметры, которые должны быть обработаны каждым транзитным узлом на пути от отправителя до получателя пакета
Routing	43	Позволяет отправителю определять список узлов, которые пакет должен пройти
Fragment	44	Содержит информацию о фрагментации пакета
Encapsulating Security Payload (ESP)	50	Обеспечивает шифрование данных с помощью IPSec
Authentication Header (AH)	51	Содержит информацию для проверки подлинности данных при использовании IPSec
Destination Options	60	Определяет произвольный набор опций, которые должны быть обработаны получателем пакета
TCP	6	Заголовок вышележащего уровня
UDP	17	
ICMPv6	58	

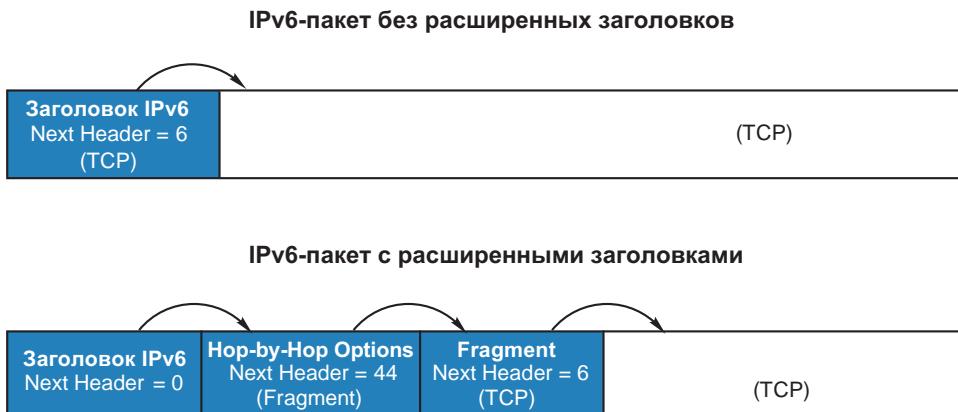
Поле Next Header используется для логической связи всех заголовков пакета IPv6. Например, Next Header в фиксированном заголовке указывает тип первого расширенного заголовка, поле Next Header в первом расширенном заголовке содержит тип следующего расширенного заголовка и т. д. Поле Next Header последнего расширенного заголовка содержит номер протокола транспортного уровня (TCP или UDP) (рис. 3.28).

Расширенные заголовки обрабатываются только узлом-получателем, за исключением заголовка *Hop-By-Hop Options*, который обрабатывается каждым промежуточным узлом на пути пакета, включая отправителя и получателя.

Каждый расширенный заголовок может появляться в дейтаграмме IPv6 только один раз (за исключением *Destination Options*, который появляется дважды). Если в дейтаграмме имеются несколько расширенных заголовков, они должны следовать в определенном порядке:

1. Hop-by-Hop Options

2. Destination Options (для опций, которые должны быть обработаны получателем, указанным в поле Destination Address плюс последовательностью получателей, перечисленных в заголовке Routing)

**Рис. 3.28.** Расширенные заголовки IPv6

3. Routing
4. Fragment
5. Authentication
6. Encapsulating Security Payload
7. Destination Options (для опций, которые должны быть обработаны только последним получателем пакета).

Подведем итог в сравнении заголовков пакетов IPv4 и IPv6:

- Неизменными остались поля *Версия* (Version), *Адрес источника* (Source Address) и *Адрес назначения* (Destination Address);
- Поле *Длина заголовка* (Internet Header Length) исчезло, так как фиксированный заголовок IPv6 имеет определенную длину (40 байт);
- Поле *Тип сервиса* (Type of Service) трансформировалось в заголовок IPv6 в поля *Класс трафика* (Traffic Class) и *Метка потока* (Flow Label);
- Поля *Время жизни* (Time to Live) и *Протокол* (Protocol) в заголовке IPv6 изменили названия, соответственно на *Предельное число шагов* (Hop Limit) и *Следующий заголовок* (Next Header) с некоторым уточнением трактовки;
- Поле *Контрольная сумма* (Header Checksum) было ликвидировано, так как ее подсчет занимает некоторое время, что существенно снижает производительность узлов;
- Поля в заголовке IPv4, связанные с фрагментацией, были перенесены в расширенные заголовки IPv6.

3.2.2. Размер пакета IPv6

Для того чтобы отправить сообщение, используя протокол IP, данные, полученные от вышестоящего уровня, инкапсулируются в IP-пакет. Далее пакет опускается на уровень ниже и инкапсулируется в кадр канального уровня, который помещается в кадр физического уровня и передается

по сети. Максимальный размер пакета, который может быть передан через физическую сеть, называется Maximum Transmission Unit (MTU).

Обычно узлы стараются отправлять большие пакеты, поскольку это уменьшает издержки — например, количество служебной информации, такой как заголовки. Когда узел отправляет IP-пакет, он должен определить, не превышает ли размер пакета MTU нижележащего протокола. Если размер пакета больше, то он разбивается на фрагменты (fragments) и каждый фрагмент отправляется в виде отдельного пакета сетевого уровня.

Пакет на своем пути от источника до приемника может пройти через множество сетей различных технологий, имеющих разные MTU. Поэтому его размер должен быть таким, чтобы вписаться в соответствующее поле протокола нижележащего уровня на каждом шаге маршрута.

В целом механизм отправки пакетов IPv6 похож на процесс отправки пакетов IPv4. Основные различия заключаются в следующем:

- Минимальный размер пакета IPv4 (фрагментированного или целого), определенный в RFC 791, был 576 байт. В пакетах IPv6 этот размер увеличился в 2 раза до 1280 байт. Это повысило эффективность передачи за счет увеличения доли содержащихся в пакете полезных данных;

- Пакеты IPv4 могли фрагментироваться или источником сообщения, или маршрутизаторами, находящимися на пути к получателю. В IPv6 фрагментировать пакеты могут только источники сообщений. Поэтому они должны выбирать такой размер фрагмента, который не превышал бы наименьший MTU на пути следования пакета. Для этого источник или использует MTU по умолчанию (1280 байт), или применяет механизм Path MTU discovery, аналогичный IPv4;

- Отражая уменьшающуюся важность фрагментации в IPv4, в IPv6 поля, связанные с фрагментацией, были перенесены из фиксированного в расширенный заголовок.

3.2.3. Представление и структура адреса IPv6

Увеличение длины адреса IPv6 до 128 бит значительно расширило доступное адресное пространство, но привело к неудобствам, связанным с его использованием. Адрес получился очень длинный, что приводит к трудностям при его запоминании и вводе. Для удобства восприятия и запоминания адреса IPv4 используется десятично-точечная нотация. Для того чтобы использовать десятично-точечную нотацию для представления адреса IPv6, его надо было бы разбить на 16 октетов и представить каждый из них десятичным числом от 0 до 255. Однако запоминать пришлось бы не 4 цифры, как в IPv4, а 16. Запись и запоминание адреса IPv6 в двоичном виде еще сложнее, так как количество цифр в нем получается равным 128.

Для того чтобы сделать адрес IPv6 короче было принято решение использовать его шестнадцатеричное представление. Шестнадцатеричная нотация адреса IPv6 похожа на представление MAC-адреса в технологии Ethernet, где

48 битов представляются 6 октетами, разделенными знаком «-» или двоеточием. Например:

Ox1A:C7:64:07:AF:D0

Так как адрес IPv6 длиннее, он записывается как *восемь групп по четырех шестнадцатеричные* цифры, разделенные двоеточием:

705B:2D9D:DC28:0000:0000:FC57:D3F8:1CAF

Существует несколько способов, которые позволяют сократить запись адреса IPv6:

- нули в начале группы можно заменить одним;
- одна или несколько идущих подряд групп, состоящих из нулей, может быть заменена знаком «::»;
- конечные нули в группе должны присутствовать.

Рассмотрим приведенный ниже адрес. Цифры, выделенные красным цветом, представляют позиции, в которых адрес может быть сокращен.

2001:1000:**0000:0000:0000**:ABCD:**0000:0001**

Варианты возможных сокращений:

2001:1000:**::**ABCD:**0:0001**

2001:1000:**::**ABCD:**0:1**

Внимание: знак «::» не может использоваться дважды, поскольку такая запись воспринимается неоднозначно. Поэтому, например, адрес 2001:1000::ABCD::1 является недействительным.

Альтернативной формой записи адреса, которая более удобна для использования в смешанной среде с узлами IPv4 и IPv6, является запись вида x:x:x:x:d.d.d.d, где x — шестнадцатеричное значение 6 первых групп адреса; d — десятичное значение 4 последних групп адреса (стандартное представление адреса IPv4). Например:

0:0:0:0:0:0:**13.1.68.3** или в сокращенном виде

::13.1.68.3

0:0:0:0:0:FFFF:**129.144.52.38** или в сокращенном виде

::FFFF:129.144.52.38

Аналогично классовым адресам IPv4, адреса IPv6 состоят из двух логических частей — битов идентификатора сети и следующих за ними битов идентификатора узла. Часть адреса, отведенная под идентификатор сети/подсети, называется *префиксом* (Prefix). Количество битов, отведенных под идентификатор сети, называется *длиной префикса* (Prefix length). Часть адреса, идентифицирующая интерфейс, называется *идентификатором интерфейса* (Interface ID). Он должен быть уникальным внутри сети/подсети.

Представление префикса адреса IPv6 аналогично записи префикса адреса IPv4 в нотации CIDR. Он записывается в виде нотации «адрес IPv6/длина префикса», где «адрес IPv6» — адрес IPv6, записанный с помощью любой формы записи, «длина префикса» — десятичное число, показывающее, сколько битов адреса отведено под префикс.

Для примера приведем запись 60-битного префикса 12AB00000000CD3 (аналогично записи номера сети/подсети в IPv4):

12AB:0000:0000:CD30:0000:0000:0000:0000/60

или

12AB::CD30:0:0:0:0/60

или

12AB:0:0:CD30::/60

Записать адрес принадлежащего подсети 12AB:0:0:CD30::/60 узла можно следующим образом:

12AB:0:0:CD30:123:4567:89AB:CDEF/60

3.2.4. Типы адресов IPv6

Адресное пространство протокола IPv6 разделено на три типа адресов:

- индивидуальные (unicast) адреса;
- групповые (multicast) адреса;
- альтернативные (anycast) адреса.

Индивидуальные адреса идентифицируют один интерфейс устройства.

Пакеты, отправленные на этот адрес, доставляются только на этот интерфейс.

Групповые адреса IPv6, подобно одноименным адресам IPv4, определяют группу интерфейсов. Пакеты, посылаемые на этот адрес, доставляются всем интерфейсам — участникам группы рассылки.

Альтернативные адреса позволяют адресовать группу интерфейсов (обычно принадлежащих разным узлам). Однако в отличие от групповых адресов, пакеты, передаваемые на альтернативный адрес, доставляются на один из интерфейсов (обычно «ближайший», согласно метрике маршрутизации), определяемых этим адресом.

Внимание: альтернативные адреса назначаются только интерфейсам маршрутизатора.

Широковещательные адреса (Broadcast), которые используются в IPv4, в IPv6 *отсутствуют*, что способствует уменьшению сетевого трафика и снижению нагрузки на большинство систем. Широковещательные адреса заменены групповыми.

Типы адресов IPv6 определяются по их битам высокого порядка как показано в табл. 3.6.

Таблица 3.6. Типы адресов IPv6

Тип адреса	Двоичный префикс	Нотация IPv6
Unspecified	00...0 (128 битов)	::/128
Loopback	00...1 (128 битов)	::1/128
Multicast	11111111	FF00::/8
Unique-Local Unicast	11111100	FC00::/7
Link-Local Unicast	111111010	FE80::/10
Global Unicast	Все остальное	

Альтернативные адреса берутся из адресного пространства индивидуальных адресов и поэтому синтаксически от них не отличаются.

3.2.5. Индивидуальные адреса

Существует несколько типов индивидуальных адресов IPv6:

- Global Unicast;
- Unique-Local Unicast;
- Link-Local Unicast.

Также имеется специальный подтип адресов Global Unicast: адреса IPv6 со встроенными адресами IPv4.

Для каждого типа индивидуального адреса определен свой диапазон (рис. 3.29).

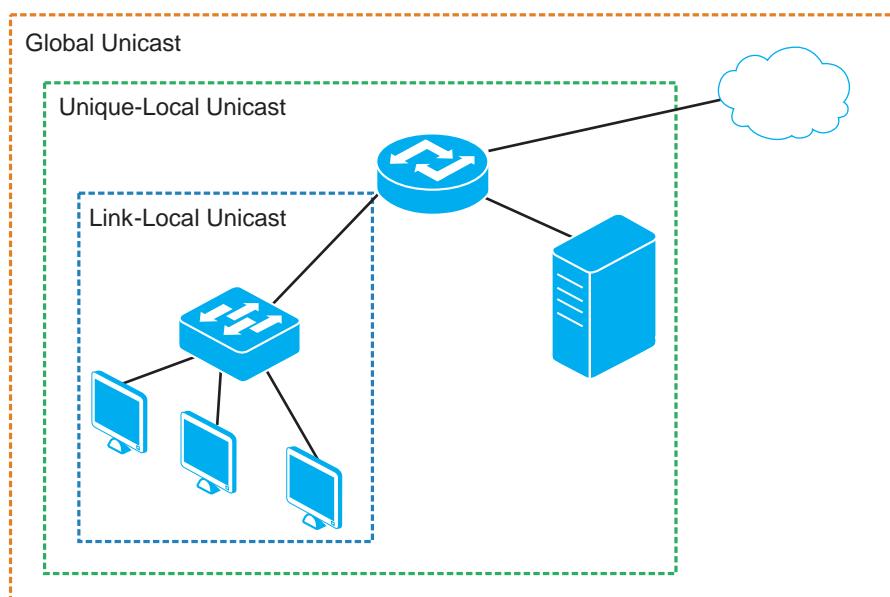


Рис. 3.29. Диапазоны индивидуальных адресов IPv6

3.2.5.1. Идентификатор интерфейса

Идентификатор интерфейса (Interface ID) в индивидуальных адресах IPv6, аналогично идентификатору узла адреса IPv4, используется для определения интерфейса в сегменте сети. Он должен быть уникальным внутри сети/подсети. При этом уникальность идентификатора интерфейса не зависит от уникальности адреса IPv6.

Для всех индивидуальных адресов, за исключением тех, которые начинаются с 000, идентификатор интерфейса должен иметь длину 64 бита и быть сформирован в соответствии с форматом Modified EUI-64.

Вместо того чтобы придумывать идентификаторы интерфейсов для узлов, их можно получать из адресов нижележащего канального уровня, так как их длина не превосходит 64 бит. В зависимости от нижележащей технологии канального уровня существуют несколько методов формирования идентификатора интерфейса в соответствии с форматом Modified EUI-64.

Наиболее распространенным типом адреса канального уровня является MAC-адрес, используемый технологиями IEEE 802. MAC-адрес, длина которого составляет 48 бит, разделен на две части. В первой части указывается уникальный *идентификатор производителя оборудования* (Organizationally Unique Identifier, OUI), который присваивается производителю институтом IEEE. Старшие 24 бита MAC-адреса, которые назначаются непосредственно производителем, и позволяют идентифицировать оборудование.

IEEE также определил формат, который называется *64-bit Extended Unique Identifier* (EUI-64). Он аналогичен формату MAC-адреса, но для идентификации оборудования используется 40 бит вместо 24 бит. Идентификатор OUI остается без изменений. Модифицированный для использования в IPv6 вариант этого формата называется модифицированным EUI-64 (Modified EUI-64).

Если на канальном уровне узел или линия связи использует адрес в формате EUI-64, то идентификатор интерфейса IPv6 получается из него очень просто. В 64-битовом адресе значение 7-го бита слева (бит “u” (universal/local)) изменяется с «0» на «1».

Большинство устройств используют MAC-адрес. Его преобразование в идентификатор интерфейса выполняется в два этапа. Сначала MAC-адрес преобразуется в формат EUI-64, а из него затем создается модифицированный EUI-64.

Рассмотрим процесс получения идентификатора интерфейса из MAC-адреса узла (рис. 3.30).

1. Преобразуем MAC-адрес в формат EUI-64. Так как MAC-адрес состоит из 48 бит, а для идентификатора интерфейса необходимо 64 бита, вставляем два октета со значениями 0xFF и 0xFE в середину MAC-адреса (между OUI и идентификатором, называемым производителем).

2. В полученном адресе EUI-64 значение бита 7 (слева) изменяем с «0» на «1».

Технологии TCP/IP в современных компьютерных сетях

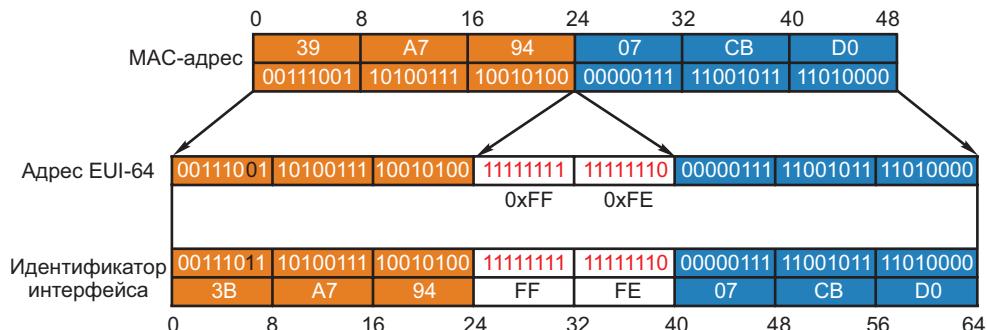


Рис. 3.30. Преобразование MAC-адреса в идентификатор интерфейса

При формировании идентификатора интерфейса из MAC-адреса существует возможность определения и отслеживания трафика конкретного узла независимо от его префикса. Для обеспечения определенного уровня анонимности в документе RFC 3041 описан метод генерации узлом псевдослучайного идентификатора интерфейса IPv6, сменяемого с течением времени (рис. 3.31).

Итоговый адрес IPv6, основанный на таком псевдослучайном идентификаторе интерфейса, называют временным адресом. Временные адреса

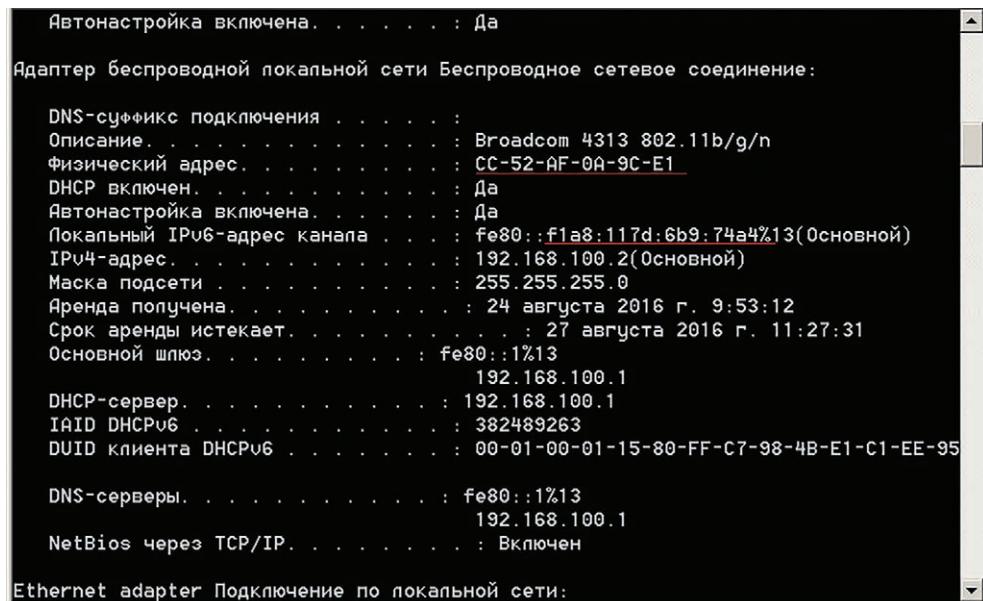


Рис. 3.31. Генерация идентификатора интерфейса на основе псевдослучайных чисел в OC Windows

рекомендуется создавать для префиксов глобальных адресов, используемых для подключения в Интернет.

По умолчанию Windows Vista, Windows 7 и выше не используют технику генерации идентификатора интерфейса из MAC-адреса. Они генерируют псевдослучайные числа для использования в качестве идентификатора интерфейса как для префиксов глобальных уникальных адресов, так и для адресов Link-Local Unicast. Адреса имеют неограниченное время жизни.

3.2.5.2. Глобальные индивидуальные адреса IPv6

Адреса IPv6 *Global Unicast* являются аналогами публичных адресов IPv4 и используются для идентификации устройств в Глобальной сети. Эти адреса выдаются IANA региональным регистраторам и имеют общий формат, показанный на рис. 3.32.

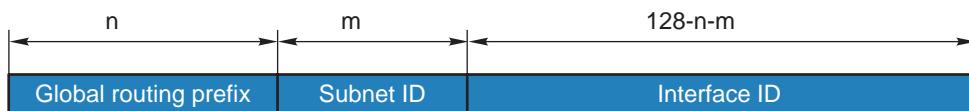


Рис. 3.32. Общий формат адресов IPv6 Global Unicast

Адрес IPv6 Global Unicast разделен на три логические части: *глобальный префикс маршрутизации* (Global routing prefix), *идентификатор подсети* (Subnet ID) и *идентификатор интерфейса* (Interface ID). У всех адресов Global Unicast, за исключением тех, которые начинаются с 000, идентификатор интерфейса должен иметь длину 64 бита и быть сформирован в соответствии с форматом Modified EUI-64. В этом случае формат адреса Global Unicast имеет вид, показанный на рис. 3.33:

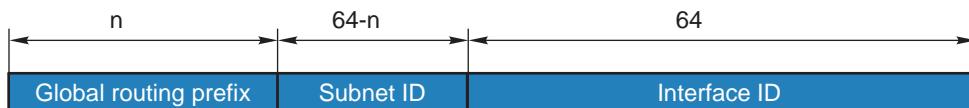


Рис. 3.33. Формат адреса IPv6 Global Unicast с идентификатором интерфейса длиной 64 бита

В настоящее время адреса IPv6 Global Unicast назначаются с префиксом 2000::/3. С учетом этого формат адреса IPv6 Global Unicast представлен на рис. 3.34.



Рис. 3.34. Формат адреса IPv6 Global Unicast с идентификатором интерфейса длиной 64 бита и префиксом 2000::/3

Адреса IPv6 Global Unicast, начинающиеся с 000, не имеют ограничений по размеру или структуре идентификатора интерфейса. Примером адресов Global Unicast, начинающихся с 000, являются адреса IPv6 со встроенными адресами IPv4. Они предназначены для использования на узлах, не поддерживающих IPv6. Было определено два типа таких адресов: IPv4-Compatible

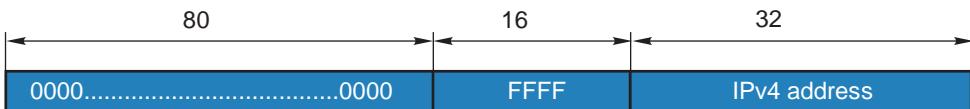


Рис. 3.35. Формат адреса IPv4-mapped IPv6

IPv6 address и IPv4-mapped IPv6 address. Адрес IPv4-Compatible IPv6 больше не используется. Адрес IPv4-mapped IPv6 (адрес IPv4, отображенный на IPv6) содержит встроенный адрес IPv4 и используется для представления адреса IPv4 как адреса IPv6. Он начинается с префикса ::FFFF:0:0/96. Его формат показан на рис. 3.35.

3.2.5.3. Локально-используемые индивидуальные адреса IPv6

Существуют два типа локально-используемых индивидуальных адресов IPv6: Unique-Local IPv6 Unicast и Link-Local IPv6 Unicast.

Адреса *Unique-Local IPv6 Unicast (ULA)*, описанные в RFC 4193, являются глобально уникальными и предназначены для адресации узлов внутри локальной сети. Пакеты, у которых в качестве источника или назначения указан этот адрес, не будут передаваться через Интернет. Такие адреса могут маршрутизироваться только внутри локальных сетей. Если провести аналогию с адресами IPv4, то адреса Unique-Local Unicast эквивалентны частным адресам IPv4, только в отличие от них являются уникальными в рамках Глобальной сети.

Формат адреса Unique-Local Unicast показан на рисунке 3.36.

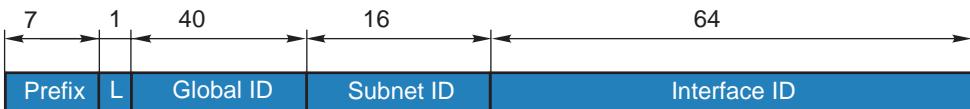


Рис. 3.36. Формат адресов Unique-Local IPv6 Unicast

Все адреса Unique-Local Unicast начинаются с префикса (Prefix) FC00::/7. Бит L показывает, что префикс назначен локально (L=1) или адрес зарезервирован для будущих применений (L=0). Таким образом, бит L разбивает префикс FC00::/7 на два поддиапазона:

- FC00::/8 — зарезервирован для будущих применений;
- FD00::/8 — локально назначенный уникальный адрес.

Следующие 40 бит отведены под *глобальный идентификатор* (*Global ID*), который определяет организацию. Он должен быть уникальным, для того чтобы минимизировать возможность совпадения с идентификаторами других организаций, поэтому назначается с помощью псевдослучайного алгоритма, который обеспечивает высокую вероятность его уникальности. Алгоритм для генерации адреса Unique-Local Unicast можно найти в Интернете (например, <https://www.ultratools.com/tools/rangeGenerator>). Далее в адресе следуют 16-битное поле *идентификатор подсети* (*Subnet ID*), которое определяет подсеть внутри сети организации, и 64-битный *идентификатор интерфейса* (*Interface ID*).

Адреса *Link-Local IPv6 Unicast* (рис. 3.31) предназначенные для взаимодействия внутри сегмента сети или по каналу связи «точка-точка», используются только в пределах данного канала. Маршрутизаторы не передают пакеты с адресами Link-Local Unicast, указанными в качестве источника или назначения, через другие линии связи. Эти адреса автоматически назначаются узлу независимо от наличия в сети маршрутизатора или DHCPv6-сервера.

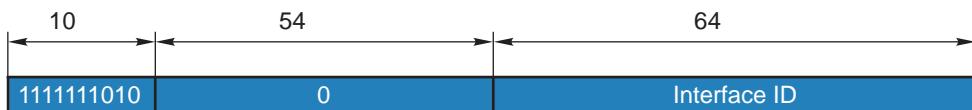


Рис. 3.37. Формат адресов Link-Local IPv6 Unicast

Адреса Link-Local Unicast имеют достаточно простой формат: адрес начинается с *глобального префикса маршрутизации* (*Global routing prefix*) FE80::/10. По сравнению с адресом Global Unicast здесь префикс стал значительно короче, поэтому пространство, отведенное под *идентификатор подсети* (*Subnet ID*), увеличилось с 16 до 54 бит. В связи с тем что адреса Link-Local Unicast используются только в пределах линии связи, поле *Subnet ID* заполняется нулями. Последние 64 бита адреса отведены под *идентификатор интерфейса* (*Interface ID*).

В IPv6, так же как и в IPv4, адрес идентифицирует не конкретное устройство, а его интерфейс. Главное отличие заключается в том, что протокол IPv6 позволяет назначить интерфейсу любое количество уникальных IPv6-адресов: только Link-Local Unicast или адреса сразу всех трех типов.

В IPv6 существуют индивидуальные адреса, используемые локально узлом. Адрес, содержащий все нули (0:0:0:0:0:0:0 или ::/128), называется *неопределенным адресом* (*Unspecified address*). Он никогда не назначается узлу и обозначает отсутствие адреса. Он обычно используется в поле «Адрес назначения» дейтаграммы IPv6, отправляемой устройством до того, как будет сконфигурирован его собственный IP-адрес. Неопределенный адрес не должен использоваться в качестве адреса назначения. Пакеты IPv6 с неопределенным адресом не маршрутизируются.

```
C:\Users\edu1>ipconfig

Настройка протокола IP для Windows

Ethernet adapter Подключение по локальной сети:

DNS-суффикс подключения . . . . . : 
IPv6-адрес . . . . . : fdd0:5f56::9055:32c6:d34b:a802
Временный IPv6-адрес . . . . . : fdd0:5f56::590e:bce2:a71:816
Локальный IPv6-адрес канала . . . . . : fe80::9055:32c6:d34b:a802%11
IPv4-адрес . . . . . : 10.90.90.100
Маска подсети . . . . . : 255.0.0.0
Основной шлюз. . . . . : fe80::222:b0ff:fe31:8800%11

Туннельный адаптер isatap.<433C9F2A-3BFF-4087-9A45-E0FA428F95FE>:

Состояние среды. . . . . : Среда передачи недоступна.
DNS-суффикс подключения . . . . . :

Туннельный адаптер Подключение по локальной сети*:

Состояние среды. . . . . : Среда передачи недоступна.
DNS-суффикс подключения . . . . . :

C:\Users\edu1>
```

Рис. 3.38. Назначенные интерфейсу адреса IPv6

Индивидуальный адрес 0:0:0:0:0:0:1 (::1/128) называется адресом «*обратной петли*» (Loopback address). Он может использоваться узлом для отправки IPv6-дейтаграммы самому себе с целью тестирования. Этот адрес не должен назначаться ни одному физическому интерфейсу и использоваться в качестве адреса источника. Он аналогичен адресу 127.0.0.1 в IPv4.

3.2.6. Альтернативные адреса

В протоколе IPv6 появился новый тип адреса — *альтернативный адрес* (IPv6 anycast address). Он назначается нескольким интерфейсам (рис. 3.38). При этом пакет, отправленный на этот адрес, направляется на «ближайший» (имеющий минимальную метрику маршрутизации) интерфейс. В соответствии с RFC 4291 альтернативный адрес не может использоваться в качестве адреса источника в пакетах IPv6 и назначается только маршрутизаторам, а не конечным узлам. Пакеты, отправленные на альтернативный адрес, будут доставлены всем маршрутизаторам сети, но данные будут передаваться только через интерфейс «ближайшего» маршрутизатора, как показано на рис. 3.39.

Альтернативным адресам не выделен специальный блок адресов, они входят в адресное пространство индивидуальных адресов. Альтернативный адрес состоит из префикса подсети (Subnet prefix), за которым следуют все 0 (рис. 3.40).

Префикс подсети может занимать столько битов, сколько необходимо для уникальной идентификации подсети, которую обслуживают маршрутизаторы.

Одним из применений альтернативных адресов является идентификация группы маршрутизаторов, принадлежащих Интернет-провайдеру. Такие адреса в маршрутном заголовке IPv6 могут использоваться в качестве

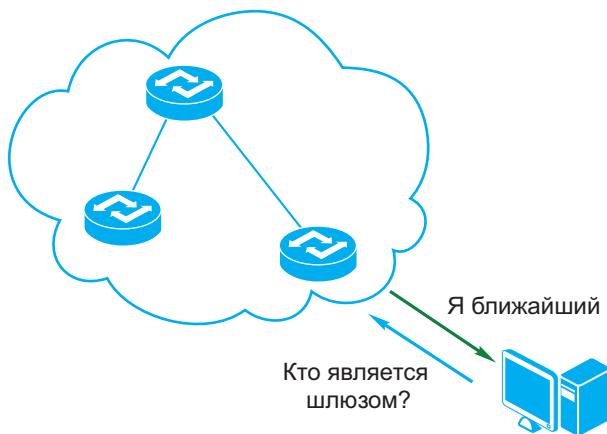


Рис. 3.39. Использование альтернативного адреса IPv6

промежуточных, чтобы обеспечить доставку пакета через определенного провайдера или последовательность провайдеров. Схема применения альтернативных адресов описана в RFC 4291.

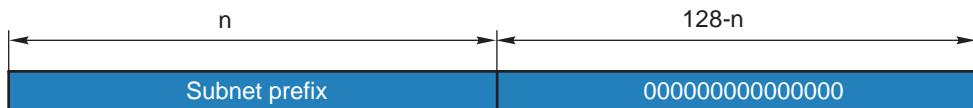


Рис. 3.40. Формат альтернативных адресов IPv6

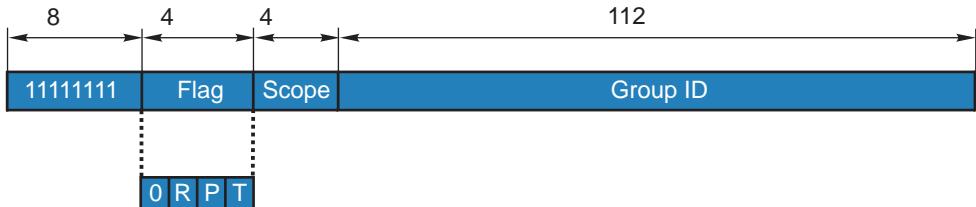
3.2.7. Групповые адреса

Многоадресная передача обеспечивает доставку потока данных группе узлов. Источник многоадресного трафика направляет пакеты многоадресной рассылки не на индивидуальные IP-адреса каждого из узлов-получателей, а на групповой IP-адрес.

Групповой адрес IPv6 (IPv6 multicast address) идентифицирует группу интерфейсов (обычно разных узлов). При этом интерфейс может входить более чем в одну группу. Групповой адрес не может использоваться в качестве адреса источника в пакетах IPv6. Формат адреса показан на рис. 3.41.

Групповые адреса начинаются с префикса FF00::/8. Следующие 4 бита — флаги (*Flag*). Первые 3 бита этого поля в настоящее время не используются и зарезервированы для будущего применения. Последний бит T определяет тип адреса:

- Т = 0 — адрес является постоянным, официально выделенным IANA для использования в Глобальной сети;
- Т = 1 — адрес является временным.

**Рис. 3.41.** Формат групповых адресов IPv6

Поле *Scope* занимает 4 бита и определяет область действия группового адреса, т. е. показывает, как далеко друг от друга могут находиться члены одной многоадресной группы. В настоящее время определены шесть значений этого поля, остальные зарезервированы для будущего применения:

- 1 — *Interface-Local* — многоадресная группа определена в рамках одного узла;
- 2 — *Link-Local* — многоадресная группа определена в пределах канала связи;
- 4 — *Admin-Local* — многоадресная группа определена внутри области, задаваемой администратором сети.
- 5 — *Site-Local* — многоадресная группа определена в рамках локальной сети;
- 8 — *Organization-Local* — многоадресная группа определена в рамках распределенной сети одной организации;
- E — *Global* — глобальная многоадресная группа.

Последние 112 битов группового адреса определяют идентификатор группы (*Group ID*) в пределах области действия адреса.

Функцию широковещательных адресов в протоколе IPv6 выполняют специальные групповые адреса, которые не могут быть назначены многоадресной группе (рис. 3.42):

- FF01::1 — идентифицирует группу, включающую в себя все IPv6-узлы в пределах области *Interface-Local*;
- FF02::1 — идентифицирует группу, включающую в себя все IPv6-узлы в пределах области *Link-Local*;
- FF01::2 — идентифицирует группу всех IPv6-маршрутизаторов в пределах области *Interface-Local*;
- FF02::2 — идентифицирует группу всех IPv6-маршрутизаторов в пределах области *Link-Local*;
- FF05::2 — идентифицирует группу всех IPv6-маршрутизаторов в пределах области *Site-Local*.

В протоколе IPv6 групповые адреса также используются в процессе разрешения адресов с помощью протокола Neighbor Discovery Protocol (NDP), т. е. получения адресов канального уровня других узлов (например, MAC-адресов) на основе известных IPv6-адресов. Адрес, который используется в процессе разрешения адресов, называется *Solicited-Node* (адрес

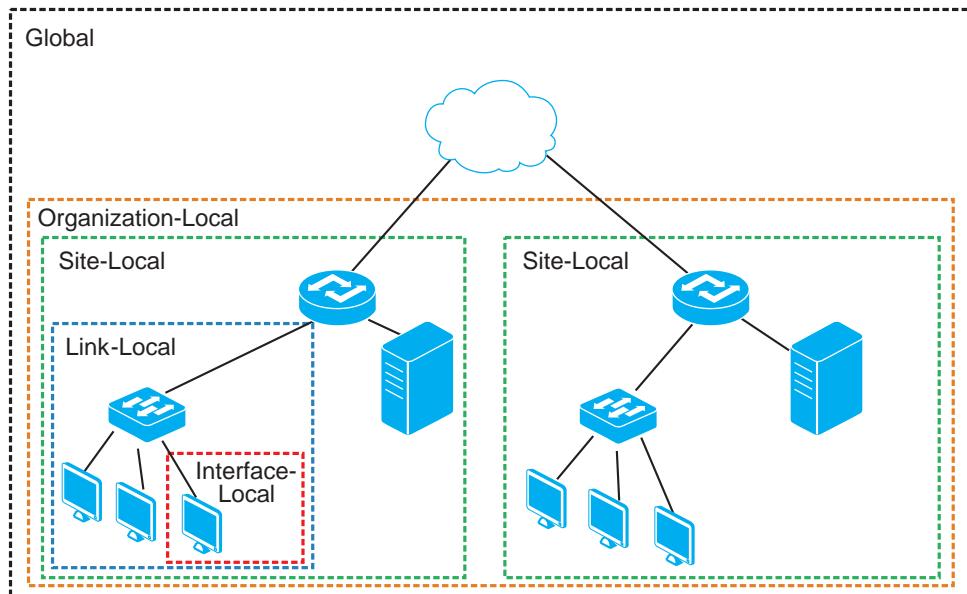


Рис. 3.42. Области действия групповых адресов IPv6

запрашивающего узла). Он должен формироваться на каждом интерфейсе для всех сконфигурированных на нем индивидуальных и альтернативных адресов. Этот адрес используется только внутри линии связи или сегмента сети.

IPv6: FE80::0202:B3FF:FE1E:8329
 Solicited-Node: FF02:0000:0000:0000:0001:FF00:0000
 Solicited-Node: FF02:0000:0000:0000:0001:FF1E:8329

FF02::1:FF1E:8329

Рис. 3.43. Формирование адреса Solicited-Node

Адрес Solicited-Node формируется из младших 24 битов поля Interface ID индивидуального или альтернативного адреса путем прибавления префикса FF02:0:0:0:1:FF00::/104 (рис. 3.43).

3.2.8. Способы конфигурации адреса IPv6

В отличие от протокола IPv4, где настройка параметров узла производится либо вручную, либо с помощью протокола DHCP, в протоколе IPv6 узел может практически самостоятельно сконфигурировать параметры своих интерфейсов.

В протоколе IPv6 определены два механизма автоконфигурации: *Stateless autoconfiguration* (описан в RFC 4862) и *Stateful autoconfiguration* (описан RFC 3315).

Stateless autoconfiguration позволяет узлам генерировать свой собственный адрес на основе комбинации локально доступной информации и информации, объявляемой маршрутизаторами. Маршрутизаторы объявляют префиксы, определяющие подсеть (или подсети), а узлы самостоятельно генерируют идентификаторы интерфейсов. В случае отсутствия маршрутизатора узлы могут автоматически генерировать адрес Link-Local IPv6 Unicast.

Stateful autoconfiguration позволяет узлам получать адрес интерфейса и конфигурационные параметры с помощью протокола DHCPv6.

Рассмотрим последовательность действий, которые выполняются в процессе *Stateless autoconfiguration*. Автоконфигурация выполняется только в том случае, если протокол канального уровня поддерживает многоадресную передачу и начинается во время процесса загрузки узла.

Шаг 1. Генерация адреса Link-Local IPv6 Unicast.

Узлы начинают процесс автоконфигурации интерфейса с генерации адреса Link-Local Unicast. Он формируется из префикса FE80::/10 и 64-битного идентификатора интерфейса.

Шаг 2. Определение дублирования адресов (Duplicate Address Detection, DAD).

Прежде чем адрес Link-Local Unicast будет присвоен интерфейсу и начнет использоваться, узел должен проверить его уникальность для данного сегмента сети. Процедура определения дублирования адресов использует сообщения Neighbor Solicitation (NS) и Neighbor Advertisement (NA) протокола Neighbor Discovery Protocol (NDP). Узел отправляет сообщение Neighbor Solicitation (NS), содержащее в качестве адреса назначения сгенерированный адрес. Если в ответ на него получено сообщение Neighbor Advertisement (NA), значит, этот адрес уже используется другим узлом. В этом случае процесс автоконфигурации завершается и требуется ручная настройка интерфейса.

Шаг 3. Присвоение адреса Link-Local IPv6 Unicast.

Если тест на уникальность успешно пройден, узел присваивает сгенерированный на первом шаге адрес Link-Local Unicast своему интерфейсу. Этот адрес может использоваться только для связи с устройствами внутри сегмента сети.

Шаг 4. Обнаружение маршрутизатора.

Следующим шагом после присвоения интерфейсу адреса Link-Local Unicast является обнаружение маршрутизатора с целью последующей генерации адресов Global и Unique-Local IPv6 Unicast. Эти адреса генерируются из префиксов, рассыпаемых маршрутизатором в объявлениях Router Advertisement (RA) (рис. 3.44), и идентификатора интерфейса, сформированного на первом шаге.

Если в сети присутствуют маршрутизаторы, они периодически рассыпают объявления на групповой адрес FF02::1, идентифицирующий все узлы

3. Протокол IP

в пределах области Link-Local. Адресом источника в сообщениях Router Advertisement (RA) является локальный адрес маршрутизатора.

Для того чтобы ускорить получение объявления от маршрутизатора, узел отправляет сообщение Router Solicitation (RS) (рис. 3.45), используя в качестве адреса источника свой адрес Link-Local Unicast, а в качестве

```
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c01, Dst: ff02::1
Internet Control Message Protocol v6
  Type: Router Advertisement (134)
  Code: 0
  Checksum: 0x1210 [correct]
  [Checksum Status: Good]
  Cur hop limit: 64
  Flags: 0x00, Prf (Default Router Preference): Medium
    0... .... = Managed address configuration: Not set
    .0... .... = Other configuration: Not set
      ..0. .... = Home Agent: Not set
      ...0 0.... = Prf (Default Router Preference): Medium (0)
      .... 0.. = Proxy: Not set
      .... .0.. = Reserved: 0
  Router lifetime (s): 1800
  Reachable time (ms): 1200000
  Retrans timer (ms): 0
  ICMPv6 Option (Source link-layer address : 84:c9:b2:1f:9c:01)
  ICMPv6 Option (Prefix information : 2012::/64)
    Type: Prefix information (3)
    Length: 4 (32 bytes)
    Prefix Length: 64
    Flag: 0xc0, On-link flag(L), Autonomous address-configuration flag(A)
    Valid Lifetime: 2592000
    Preferred Lifetime: 604800
    Reserved
    Prefix: 2012::
```

Рис. 3.44. Сообщение Router Advertisement

```
Internet Protocol Version 6, Src: fe80::cc7f:2dc1:4623:249a, Dst: ff02::2
Internet Control Message Protocol v6
  Type: Router Solicitation (133)
  Code: 0
  Checksum: 0xc6d0 [correct]
  [Checksum Status: Good]
  Reserved: 00000000
  ICMPv6 Option (Source link-layer address : 20:6a:8a:72:a5:82)
    Type: Source link-layer address (1)
    Length: 1 (8 bytes)
    Link-layer address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
```

Рис. 3.45. Сообщение Router Solicitation

Технологии TCP/IP в современных компьютерных сетях

адреса получателя — адрес группы всех маршрутизаторов в сегменте сети FF02::2.

Если окажется, что в сети нет маршрутизатора, то узел должен попытаться использовать Stateful DHCPv6 для получения адреса и другой конфигурационной информации. Следует отметить, что узел может одновременно использовать механизмы Stateless autoconfiguration и Stateless DHCPv6. Механизм Stateless DHCPv6 служит для получения дополнительных конфигурационных параметров, таких как адреса DNS-, SIP-серверов.

В сообщении Router Advertisement (RA) имеются флаги *M* (Managed Address Configuration Flag) и *O* (Other Configuration Flag). С их помощью маршрутизатор сообщает узлам, автоконфигурацию какого типа выполнять: продолжать Stateless autoconfiguration или использовать Stateful DHCPv6 для получения IPv6-адреса (рис. 3.46).

Если флаг *M* (длина 1 бит) установлен, клиент должен получить адрес IPv6 и другие конфигурационные параметры через Stateful DHCPv6.

Если флаг *O* (длина 1 бит) установлен, клиент должен получить дополнительные конфигурационные параметры (но не адрес) через Stateless DHCPv6.

Если флаги *M* и *O* не установлены, клиент выполняет только Stateless autoconfiguration.

Шаг 5. Генерация адресов Global и Unique-Local IPv6 Unicast.

В зависимости от настроек интерфейса маршрутизатора, к которому подключен узел, объявления Router Advertisement (RA) могут содержать

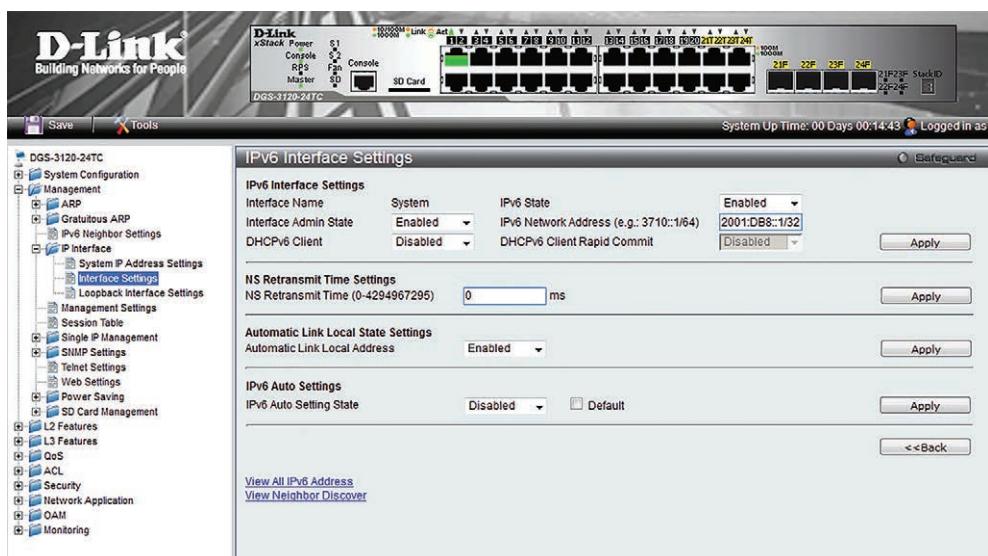


Рис. 3.46. Пример настройки интерфейса IPv6 маршрутизирующего устройства для рассылки объявлений RA

информацию о префиксах для генерации как обоих адресов Global и Unique-Local IPv6 Unicast, так и какого-то одного из них.

В любом случае при Stateless autoconfiguration адрес будет формироваться из префикса, предоставленного маршрутизатором, и идентификатора интерфейса, созданного на шаге 1. На рис. 3.47 показан процесс формирования адреса IPv6 Global Unicast.

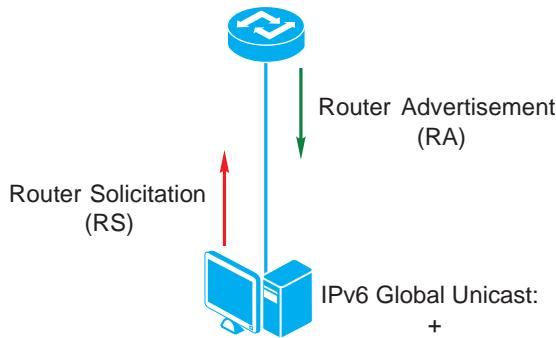


Рис. 3.47. Генерация адреса IPv6 Global Unicast при использовании механизма Stateless autoconfiguration

В протоколе IPv6, так же как и в протоколе IPv4, существует возможность ручной настройки на интерфейсе IPv6-адреса, длины префикса, шлюза по умолчанию. Ручная настройка обычно используется для конфигурации интерфейсов маршрутизаторов. Если в сети нет маршрутизирующих устройств, которые рассыпают объявления с информацией, требуемой для автоматической конфигурации узла или DHCPv6-сервера, интерфейс узла может быть настроен вручную.

Процесс автоконфигурации с использованием протокола DHCPv6 будет рассмотрен в гл. 9.

3.2.9. Планирование подсетей IPv6

Адресное пространство IPv6 позволяет гибко планировать схему адресации сети. Документы RFC для IPv6 рекомендуют использовать префикс длиной 64 бита для адресации узлов в локальных сетях. Рассмотрим пример планирования подсетей IPv6.

Предположим, что организация планирует использовать в своей сети адреса Unique-Local Unicast и хочет разбить сеть на пять подсетей. Процесс начинается с формирования 64-битного префикса сети. Напомним, что адреса Unique-Local Unicast начинаются с префикса FD00::/8. Следующий за префиксом идентификатор Global ID (40 бит) получаем с помощью генератора адресов Unique-Local IPv6 Unicast. Например, с помощью генератора

получили Global ID, равный 895a473947. Затем назначаются пять номеров подсети (Subnet ID) длиной 16 бит. Эти номера можно придумать самостоятельно или воспользоваться генератором. После того как получены все значения, объединяем их и формируем пять префиксов подсетей длиной 64 бита (табл. 3.7). Внутри каждой из полученных подсетей можно адресовать 264 узлов.

Таблица 3.7. Формирование подсетей IPv6

Prefix/L	Global ID	Subnet ID	Объединенный префикс подсети	Диапазоны IP-адресов
fd	895a473947	0710	fd89:5a47:3947:0710::/64	IP-адрес: fd89:5a47:3947:710:xxxx:xxxx:xxxx:xxxx Диапазон адресов: fd89:5a47:3947:710:0:0:0:0 — fd89:5a47:3947:710:ffff:ffff:ffff:ffff Количество узлов: 18446744073709551616
		0711	fd89:5a47:3947:0711::/64	IP-адрес: fd89:5a47:3947:711:xxxx:xxxx:xxxx:xxxx Диапазон адресов: fd89:5a47:3947:711:0:0:0:0 — fd89:5a47:3947:711:ffff:ffff:ffff:ffff Количество узлов: 18446744073709551616
		0712	fd89:5a47:3947:0712::/64	IP-адрес: fd89:5a47:3947:712:xxxx:xxxx:xxxx:xxxx Диапазон адресов: fd89:5a47:3947:712:0:0:0:0 — fd89:5a47:3947:712:ffff:ffff:ffff:ffff Количество узлов: 18446744073709551616
		0713	fd89:5a47:3947:0713::/64	IP-адрес: fd89:5a47:3947:713:xxxx:xxxx:xxxx:xxxx Диапазон адресов: fd89:5a47:3947:713:0:0:0:0 — fd89:5a47:3947:713:ffff:ffff:ffff:ffff Количество узлов: 18446744073709551616
		0714	fd89:5a47:3947:0714::/64	IP-адрес: fd89:5a47:3947:714:xxxx:xxxx:xxxx:xxxx Диапазон адресов: fd89:5a47:3947:714:0:0:0:0 — fd89:5a47:3947:714:ffff:ffff:ffff:ffff Количество узлов: 18446744073709551616

3.3. Обзор архитектуры безопасности для протокола IP

Одним из недостатков первоначального протокола IP было отсутствие каких-либо механизмов, обеспечивающих аутентификацию и целостность данных, передаваемых через составную сеть. Для обеспечения различных сервисов безопасности на уровне IP для протоколов IPv4 и IPv6 был разработан набор протоколов *IP Security* или *IPSec*.

IPSec — это набор открытых стандартов для обеспечения частных коммуникаций через IP-сети. В зависимости от того как IPSec реализован и сконфигурирован, он может предоставлять любые комбинации из следующих видов защиты:

- **Конфиденциальность:** IPSec может гарантировать, что данные не будут прочитаны неавторизованными сторонами. Это достигается путем шифрования данных с использованием криптографических алгоритмов и секретных ключей. Данные могут быть расшифрованы только с помощью секретных ключей;

- **Целостность:** IPSec может определять, была ли информация изменена в процессе передачи. Целостность данных обеспечивается генерацией кода идентификации сообщения (message authentication code, MAC), который является криптографической контрольной суммой данных. Если данные были изменены, значения MAC, вычисленные на стороне приемника и передатчика, будут различаться;

- **Взаимная аутентификация:** каждая конечная точка IPSec подтверждает идентификационную информацию другой конечной точки IPSec, с которой хочет обмениваться информацией. Это гарантирует, что сетевой трафик и данные отправлены ожидаемым узлом;

- **Защита от replay-атак:** одна и та же информация не доставляется множество раз, и данные не доставляются не по порядку. Однако IPSec не гарантирует, что данные доставляются строго в том порядке, в котором были отправлены;

- **Управление доступом:** конечные точки IPSec могут выполнять фильтрацию, гарантируя, что только авторизованные IPSec пользователи могут получать доступ к определенным сетевым ресурсам. Конечные точки IPSec также разрешают или блокируют прохождение определенного типа сетевого трафика.

Перечисленные сервисы предоставляются на уровне IP, обеспечивая защиту для самого протокола IP и всех протоколов более высокого уровня, которые могут передаваться по IP.

В большинстве случаев реализации IPSec используются для обеспечения сервисов виртуальных частных сетей (VPN). С помощью протоколов IPSec можно реализовать различные топологии VPN, основными из которых являются следующие:

- Шлюз безопасности — шлюз безопасности (Security gateway — Security gateway);
 - Узел — шлюз безопасности (Host — Security gateway);
 - Узел — узел (Host — Host).

Термин «шлюз безопасности» (security gateway) используется в документах, описывающих IPSec для обозначения промежуточной системы, которая реализует протокол, например, маршрутизатор, межсетевой экран или сервер.

VPN на основе IPSec часто используются для обеспечения безопасной передачи данных между двумя сетями, например, для соединения главного офиса и филиала через Интернет. Это обычно делается путем установки в каждой из сетей шлюзов безопасности и создания между двумя шлюзами VPN-подключения (рис. 3.48). Шлюзом может быть как выделенное устройство, выполняющее только функции VPN, так и межсетевой экран, маршрутизатор или сервер. Чтобы инициировать подключение, один из шлюзов отправляет другому запрос на установку IPSec-соединения. Два шлюза обмениваются друг с другом информацией и создают IPSec-соединение. В каждой из соединяемых сетей настраивается маршрутизация, так как узлам из разных сетей необходимо взаимодействовать друг с другом. Их трафик будет автоматически маршрутизироваться через IPSec-соединение и соответствующим образом защищаться.

Между двумя шлюзами может быть установлено единственное IPSec-соединение, защищающее все виды трафика, или множество IPSec-соединений, которые могут защищать различные типы или классы трафика (в случае QoS).

Задача соединений между узлами и локальными шлюзами обычно не выполняется. Для пользователей эта топология прозрачна и не требует установки или настройки никакого специального программного обеспечения на компьютерах или серверах.



Рис. 3.48. Пример топологии шлюзов безопасности — шлюз безопасности

В последние годы получила широкое распространение модель узел — шлюз безопасности, которая используется для обеспечения удаленного доступа пользователей через Интернет. Организация устанавливает в своей сети VPN-шлюз; каждый удаленный пользователь создает VPN-подключение между своим локальным компьютером (узлом) и шлюзом безопасности организации. Шлюзом, как и в предыдущем случае, может быть выделенное

устройство, выполняющее только функции VPN, межсетевой экран или маршрутизатор.

В этой модели IPSec-соединение создается для каждого отдельного удаленного пользователя. Устройства удаленных пользователей должны быть настроены на работу в качестве IPSec-клиента (на них должно быть установлено программное обеспечение клиента IPSec). Когда удаленный пользователь хочет получить доступ к ресурсам организации через VPN, узел инициирует обмен данными со шлюзом безопасности. Прежде чем установить соединение шлюз выполняет аутентификацию пользователя самостоятельно или с использованием внешних серверов аутентификации. Клиент и шлюз обмениваются информацией и если аутентификация успешна, IPSec-соединение устанавливается. После этого пользователь может получить доступ к ресурсам организации. Трафик, передаваемый между узлом и шлюзом, защищается IPSec. Трафик между пользователем и системами, не контролируемыми организацией, также может маршрутизироваться через шлюз безопасности. Этот трафик, если необходимо, тоже может защищаться с помощью IPSec.

Топология узел — шлюз безопасности не прозрачна для пользователей, т.к. каждый пользователь должен быть аутентифицирован перед использованием VPN-соединения и на каждом узле должно быть установлено и настроено программное обеспечение IPSec-клиента (рис. 3.49).



Рис. 3.49. Пример топологии узел — шлюз безопасности

Последняя топология VPN — модель узел — узел (рис. 3.50). Она обычно используется для специальных целей, когда, например, сетевой администратор выполняет удаленное управление одним сервером. В этом случае организация настраивает сервер для обеспечения сервисов IPSec, а компьютер системного администратора исполняет роль клиента. Администратор использует IPSec-клиент, когда к удаленному серверу необходимо установить защищенное соединение.

В этой модели IPSec-соединение создается по желанию каждого отдельного пользователя. Когда пользователь хочет получить доступ к ресурсам

сервера, он инициирует взаимодействие с ним. Прежде чем установить соединение, сервер выполнят аутентификацию пользователя. Пользователь также может аутентифицировать сервер. Клиент и сервер обмениваются информацией, и если аутентификация успешна, IPSec-соединение устанавливается. После этого трафик между узлом пользователя и сервером будет передаваться по защищенному соединению.



Рис. 3.50. Пример топологии узел – узел

IPSec является сложной темой, поэтому в данном курсе будет дан только обзор архитектуры безопасности для протокола IP. Подробно IPSec описан в курсе «Основы сетевой безопасности. Часть 2: Технологии туннелирования», доступном для изучения на портале дистанционного обучения и сертификации D-Link.

Способы реализации IPSec

Существуют несколько способов реализации IPSec на узле, маршрутизаторе или межсетевом экране (для создания шлюза безопасности).

1. Интеграция IPSec в конкретную реализацию протокола IP. Это требует доступа к исходному коду IP и делается как на узлах, так и на шлюзах безопасности.

2. Реализация «Bump-in-the-stack» (BITS). IPSec создает новый архитектурный уровень, встраивая свою реализацию между стандартной реализацией IP-протоколов (IP-уровнем) и локальными сетевыми драйверами (канальным уровнем). Доступа к исходному коду стека IP в данном случае не требуется. Данный подход обычно применяется на узлах, когда IPSec реализован в виде подключаемой библиотеки.

3. Использование внешнего (аппаратного) криптопроцессора. Обычно это называется реализацией «Bump-in-the-wire» (BITW). Такие реализации могут использоваться как на узлах, так и на шлюзах безопасности. Обычно BITW-устройства являются IP-адресуемыми.

3.3.1. Компоненты IPSec

Несколько RFC описывают основные компоненты IPSec (табл. 3.8). Эти компоненты и их взаимодействие определяют логическую архитектуру IPSec.

Таблица 3.8. Стандарты IPSec

Номер RFC	Название
4301	Security Architecture for the Internet Protocol
4302	IP Authentication Header
4303	IP Encapsulating Security Payload (ESP)
8221	Cryptographic Algorithm Implementation Requirements For Encapsulating Security Payload (ESP) and Authentication Header (AH)
2408	Internet Security Association and Key Management Protocol (ISAKMP)
2409	The Internet Key Exchange (IKE)
7296	The Internet Key Exchange (IKEv2) Protocol
2412	The OAKLEY Key Determination Protocol

В архитектуре IPSec можно выделить четыре основные части:

- Безопасная ассоциация;
- Протоколы IPSec;
- Алгоритмы и методы шифрования/хеширования;
- Безопасная ассоциация и управление ключами.

Безопасная ассоциация

Безопасная ассоциация (Security Association, SA) представляет собой сложное (однонаправленное) логическое соединение между двумя конечными точками IPSec, служащее для предоставления сервисов безопасности передаваемому через него трафику. Другими словами, SA — это набор информации о безопасности (алгоритмов и параметров, таких как ключи шифрования), который описывает, как шифровать и аутентифицировать поток данных между двумя устройствами (рис. 3.51). Набор реализуемых SA-сервисов зависит от выбранного протокола IPSec, режима его работы, дополнительных сервисов и конечных точек SA.

При обычном двунаправленном соединении между двумя устройствами создаются две SA (по одной на каждое направление).

Для предоставления сервисов используется один из протоколов IPSec: Authentication Header (AH) или Encapsulating Security Payload (ESP). Одновременно на одной SA они использоваться не могут. В этом случае создаются две отдельные SA. Например, если для IPSec-соединения между двумя устройствами требуется использовать оба протокола IPSec, будет создано четыре SA.

SA может переносить как одноадресный, так и многоадресный трафик. В этом разделе будем рассматривать SA только для одноадресных соединений.

Весь трафик, передаваемый по SA, обрабатывается в соответствии с политикой безопасности, заданной на концах соединения. Для обеспечения защищенного соединения SA требует наличия двух баз данных: базы данных политики безопасности (Security Policy Database, SPD) и базы данных безопасных ассоциаций (Security Association Database, SAD).

```
Device:/> ipsec -show -usage

--- Active IPsec SAs:

1 IPsec tunnel      : ipsec
  Local network     : 192.168.12.0/24
  Remote network    : 172.17.100.0/24
  Remote endpoint   : 192.168.20.20
  Protocol          : ESP: DES HMAC-SHA1
  SPI (in)          : 0x3549cca2
  SPI (out)         : 0x24aba78b

  Counters:
  Packets in        : 129
  Packets out       : 121
  Kilobytes in      : 9
  Kilobytes out     : 14
  Dropped packets   : 0
  Failed MAC         : 0

1 of 1 active SAs displayed
```

Рис. 3.51. Пример SA

База данных политики безопасности (Security Policy Database, SPD) определяет набор правил, описывающих как обрабатывать входящие (inbound) и исходящие (outbound) IP-дейтаграммы (рис. 3.52). Для любой исходящей или входящей дейтаграммы существуют три возможных способа обработки: отбросить дейтаграмму, обойти IPSec или применить IPSec. Первый вариант означает, что трафик не разрешен для узла, не может проходить через шлюз безопасности или не может быть доставлен на уровень приложений. Второй вариант означает, что трафику разрешено проходить без какой-либо обработки IPSec. Третий вариант означает, что к трафику применяется IPSec-защита, и для такого трафика в SPD должны быть указаны необходимые сервисы безопасности, которые содержат информацию о применяемых протоколах, алгоритмах и т. д.

При реализации IPSec должна быть создана как минимум одна база SPD. Она представляет собой упорядоченную базу данных (упорядоченный список записей политики), согласующуюся с использованием списков управления доступом (Access Control List, ACL) или пакетными фильтрами. Создает SPD пользователь или администратор сети при настройке устройства. Так как шлюзы безопасности зачастую обладают функциональностью IPSec и межсетевого экрана, правила фильтрации могут создаваться как для IPSec, так и не-IPSec-трафика.

Каждое правило в SPD идентифицируется одним или несколькими *селекторами*, которые определяют IP-трафик, попадающий под действие этого правила. Возможные селекторы являются набором информации, доступной в заголовках IP и транспортного уровня. Во всех реализациях IPSec поддерживаются следующие параметры селекторов:

#	Name	Source	Destination	Action	Log	Usage
				Service		
1	ping_fw	lan1:192.168.10.0/24	core:192.168.10.1	Allow	Yes	1
2	drop_smb-all	lan1:192.168.10.0/24	wan1:0.0.0.0/0	Drop	Yes	0
3	allow_ping-outbound	lan1:192.168.10.0/24	wan1:0.0.0.0/0	NAT	Yes	0
4	allow_ftp	lan1:192.168.10.0/24	wan1:0.0.0.0/0	NAT	Yes	0
5	allow_standard	lan1:192.168.10.0/24	wan1:0.0.0.0/0	ftp	NAT	51
6	to_vpn	lan3:192.168.12.0/24	ipsec:172.17.100.0/24	all_tcpudp	Allow	11
7	from_vpn	ipsec:172.17.100.0/24	lan3:192.168.12.0/24	all_services	Allow	10
				all_services		

Рис. 3.52. Пример SPD

- Remote IP Address (IPv4 или IPv6): список диапазонов адресов IPv4 или IPv6 удаленных устройств. Эта структура позволяет определить один IP-адрес или диапазон IP-адресов;
- Local IP Address (IPv4 или IPv6): список диапазонов адресов IPv4 или IPv6 локальных устройств. Эта структура позволяет определить один IP-адрес или диапазон IP-адресов;
- Next Layer Protocol: индивидуальный номер порта, который определяет приложение;
- Name: имя может использоваться как символьный идентификатор для локального или удаленного IP-адреса. В отличие от других селекторов, имя не берется из пакета.

В зависимости от используемого селектора SA может быть определена детально или грубо. Например, для передачи всего трафика между двумя узлами может быть создана единственная SA и к нему может применяться единый набор сервисов безопасности. В другом случае трафик между парой узлов может передаваться через несколько SA, которые создаются для отдельных приложений. Различные SA будут предлагать различные сервисы безопасности. Это может определяться требованиями к безопасности конкретных приложений. Аналогично, если весь трафик между двумя шлюзами безопасности может передаваться через единственную SA, или для каждой пары узлов, расположенных за шлюзом безопасности, может быть определена отдельная SA.

Для каждого входящего и исходящего пакета с помощью селектора происходит поиск соответствующей ему записи в SPD (рис. 3.53). Каждая запись SPD указывает действие, в соответствии с которым пакет должен быть пропущен, отброшен или обработан IPSec. Если применяется обработка IPSec, запись содержит информацию о режиме работы, применяемом протоколе

IPSec и алгоритмы, конечных точках туннеля (для туннельного режима) и необходимые дополнительные параметры.

База данных безопасных ассоциаций (Security Association Database, SAD) хранит параметры каждой активной SA. Записи в SAD автоматически создаются протоколом Internet Key Exchange (IKE) или вручную. Для исходящей обработки записи в SAD связаны с записями в SPD. Если при обработке очередного исходящего пакета обнаружено, что запись SPD не ссылается на SA в SAD (SA не найдена в SAD), протокол IKE создает SA и помещает о ней запись в SAD. После этого запись в SPD связывается с записью в SAD.

```
Device:/> ipsec
--- Active IPsec SAs:
IPsec Tunnel      Local Network      Remote Network      Remote Endpoint
-----           -----
ipsec            192.168.12.0/24    172.17.100.0/24   192.168.20.20
1 of 1 active SAs displayed
```

Рис. 3.53. Пример SAD

Для IPSec-обработки в SAD должна присутствовать следующая информация:

- *Security Parameters Index (SPI)*: 32-битное значение, выбираемое принимающей стороной SA для ее уникальной идентификации. При обработке исходящего трафика SPI используется для создания заголовка AH или ESP. Для обработки входящего трафика SPI из заголовка AH или ESP используется для поиска соответствующей SA в SAD;
 - *Sequence Number Counter*: 64-битное значение, используемое для создания поля Sequence Number в заголовках AH или ESP для исходящего трафика;
 - *Sequence Number Overflow*: флаг, указывающий, было ли переполнение Sequence Number Counter; должен вызывать событие аудита и предотвращать передачу дополнительных пакетов по данной SA (используется только для исходящего трафика);
 - Алгоритм обеспечения целостности, ключи и параметры;
 - Алгоритм шифрования, ключи, режим, вектор инициализации IV и т. д.;
 - Время жизни данной SA: интервал времени, после которого SA должна быть заменена на новую или завершена. Это может зависеть от времени существования SA или количества байтов, переданных по SA.
- Для поиска соответствующей SA в SAD используются комбинации следующих параметров:
- Security Parameters Index (SPI);
 - IP-адреса назначения;
 - Идентификатора протокола IPSec.

Безопасная ассоциация и управление ключами

IPSec предполагает возможность как автоматического создания SA, так и создания SA вручную.

Распределение ключей вручную является простейшим способом создания SA, определения алгоритмов и ключей, при котором администратор вручную конфигурирует каждую систему, указывая ключи и другие параметры. Такая технология применяется в маленьких статичных окружениях и не масштабируется. Если количество узлов мало и если все узлы расположены в пределах одного административного домена, то возможно применение ручных технологий распределения ключей. В данном случае трафик будет защищаться с использованием вручную сконфигурированных ключей.

Для широкого использования IPSec был разработан протокол Internet Key Exchange (IKE). IKE является гибридным протоколом, который объединяет функции трех других протоколов (рис. 3.54). Первым из них является протокол Internet Security Association and Key Management Protocol (ISAKMP). Он определяет форматы пакетов для ведения переговоров об установлении, изменении и удалении SA. Эти форматы обеспечивают основу для совместимости оборудования с поддержкой IPSec разных производителей.

Протокол ISAKMP не выполняет непосредственный обмен ключами. Он позволяет использовать несколько протоколов обмена ключами внутри одной структуры. Внутри структуры, стандартизованной для IPSec, ISAKMP связывает части протоколов обмена ключами SKEME и Oakley.

Большинство процессов обмена ключами протокола IKE основаны на протоколе Oakley. Этот протокол позволяет двум аутентифицированным сторонам договориться о безопасной и секретной информации для создания ключей шифрования. Oakley основан на алгоритме обмена ключами Диффи — Хеллмана (Diffie — Hellman). Алгоритм Диффи — Хеллмана может использовать один из номеров групп, которые определяют длину ключей, создаваемых в процессе обмена ключами. Чем больше номер, тем выше стойкость ключа.

В протоколе Oakley определено несколько режимов обмена ключами. Эти режимы соответствуют двум фазам переговоров, определенным в протоколе ISAKMP. В первой фазе протокол Oakley определяет два режима работы: основной (main) и агрессивный (aggressive). Во второй фазе Oakley определяет один режим работы: быстрый (quick) режим.

При использовании с IPSec протокол IKE ассоциируется с IPsec Domain of Interpretation (DOI). Понятие домена IPSec (DOI) вводится для того, чтобы можно было сгруппировать относящиеся к IPSec протоколы, использующие IKE для ведения переговоров о SA. Протоколы безопасности,

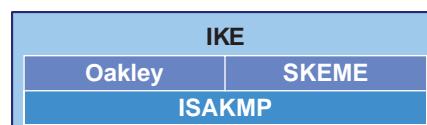


Рис. 3.54. Взаимосвязь между протоколами IKE, ISAKMP, Oakley и SKEME

относящиеся к одному DOI-домену, выбирают протокол безопасности и криптографические преобразования из общего пространства имен и используют общие идентификаторы в протоколе создания SA. Они также одинаково интерпретируют данные, содержащиеся в различных сообщениях.

Протоколы IPSec

Сервисы безопасности трафика IPSec реализуются с использованием протоколов *Authentication Header (AH)* и *Encapsulating Security Payload (ESP)*. Протокол Authentication Header (AH) обеспечивает аутентификацию источника данных и целостность сообщения, дополнительно поддерживая защиту от replay-атак. Протокол Encapsulating Security Payload (ESP) предлагает аналогичные сервисы, к которым добавляется конфиденциальность. Оба протокола обеспечивают контроль доступа, усиленный через механизм распределения ключей шифрования (протоколы Internet Key Exchange (IKE) версии 1 и 2) и управление потоками трафика в соответствии с базой данных политики безопасности (Security Policy Database).

IPSecv3 (RFC 4301) требует, чтобы реализация ESP была обязательной. Протокол AH может поддерживаться, но это не обязательно. При использовании протокола ESP без функций конфиденциальности он обеспечивает аналогичные AH сервисы. Протоколы ESP и AH могут использоваться как отдельно друг от друга, так и вместе. Однако в большинстве случаев используется только ESP.

Каждый протокол поддерживает два режима работы: *транспортный* (transport mode) и *туннельный* (tunnel mode).

Протоколы ESP и AH обеспечивают защиту, добавляя к дейтаграмме заголовок (и, возможно, другие поля), содержащий информацию о защите. Выбор режима работы не влияет на метод, с помощью которого эти заголовки генерируются, но влияет на то, какие части IP-дейтаграммы защищаются и как заголовки ESP и AH располагаются для выполнения этого. Режим является основой для создания безопасной ассоциации (SA).

Транспортный режим обычно используется для создания VPN между двумя узлами. Как следует из названия режима, он защищает сообщение, передаваемое с транспортного уровня на уровень IP. В IPv4 заголовок протокола ESP или AH добавляется перед заголовком протокола вышестоящего уровня (TCP или UDP), т. е. сразу после IP-заголовка. В IPv6 заголовок протокола ESP или AH добавляется перед заголовком протокола вышестоящего уровня, т. е. сразу за фиксированным заголовком и выбранными расширенными заголовками. Расширенный заголовок Destination options может быть как до, так и после заголовка протокола безопасности.

В протоколе ESP транспортный режим обеспечивает сервисы безопасности только для протоколов вышестоящего уровня (нагрузки IP-пакета). В случае AH защита распространяется также и на отдельные части IP-заголовка, расширенных заголовков и опций, предшествующих заголовку протокола безопасности.

Туннельный режим полностью защищает IP-дейтаграмму (с заголовком и данными), формируя виртуальный туннель между устройствами с поддержкой IPSec. Заголовок протокола безопасности добавляется перед внутренним IP-заголовком, т. е. перед IP-заголовком оригинальной дейтаграммы, который содержит адреса ее непосредственного источника и приемника. Перед заголовком протокола безопасности добавляется внешний IP-заголовок, адресами источника и приемника в котором являются адреса шлюзов безопасности, обрабатывающих пакет. Таким образом, оригинальная IP-дейтаграмма защищается и инкапсулируется в другую IP-дейтаграмму.

При использовании AH в туннельном режиме защищаются весь туннелируемый IP-пакет и части внешнего IP-заголовка. Если применяется ESP, защита обеспечивается только для туннелируемого IP-пакета.

Если одним из концов соединения является шлюз безопасности, то SA по стандартам IPSec обязательно должна выполняться в туннельном режиме, но многие производители допускают в этом случае как туннельный, так и транспортный режим. Заметим, что когда трафик предназначен для шлюза безопасности, например, в случае ping- или SNMP-команд, то шлюз безопасности рассматривается как узел и, как правило, используется транспортный режим. Два узла могут при необходимости устанавливать туннельный режим.

Подведем краткие итоги.

1. Узел должен поддерживать транспортный и туннельный режимы.

2. Шлюз безопасности должен поддерживать туннельный режим. Если он также поддерживает транспортный режим, то этот режим используется только тогда, когда шлюз рассматривается как узел.

Алгоритмы и методы шифрования/хеширования

Протоколы AH и ESP используют алгоритмы HMAC (Hash-based Message Authentication Code, код аутентификации (проверки подлинности) сообщений, использующий хеш-функции) для обеспечения целостности сообщений. В IPSecv3 определено использование алгоритмов HMAC-MD5-96, HMAC-SHA1-96, DES-MAC, KPDK-MD5, AES-XCBC-96 (рис. 3.55).

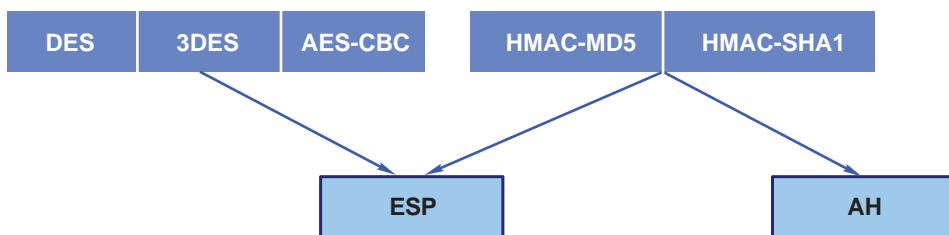


Рис. 3.55. Алгоритмы шифрования и аутентификации

Протокол ESP предоставляет возможность шифрования пакетов. Совместно с ним могут использоваться алгоритмы DES, 3DES, AES-CBC, AES-CTR, RC4, RC5, CAST, BLOWFISH и др.

Методы аутентификации IPSec, как определено в IKE, делятся на три группы: цифровая подпись (digital signature), аутентификация с открытым ключом (public key) и аутентификация на основе предварительно установленных ключей (pre-shared key).

Реализация IKE должна по умолчанию поддерживать следующие алгоритмы:

- алгоритм шифрования DES;
- хеш-функции MD5 и SHA1;
- аутентификацию на основе предварительно установленных ключей.

Это требование обусловлено необходимостью обеспечения совместимости оборудования разных производителей между собой.

Обработка IP-трафика

На рис. 3.56 показаны описанные ранее компоненты IPSec и их взаимодействие.

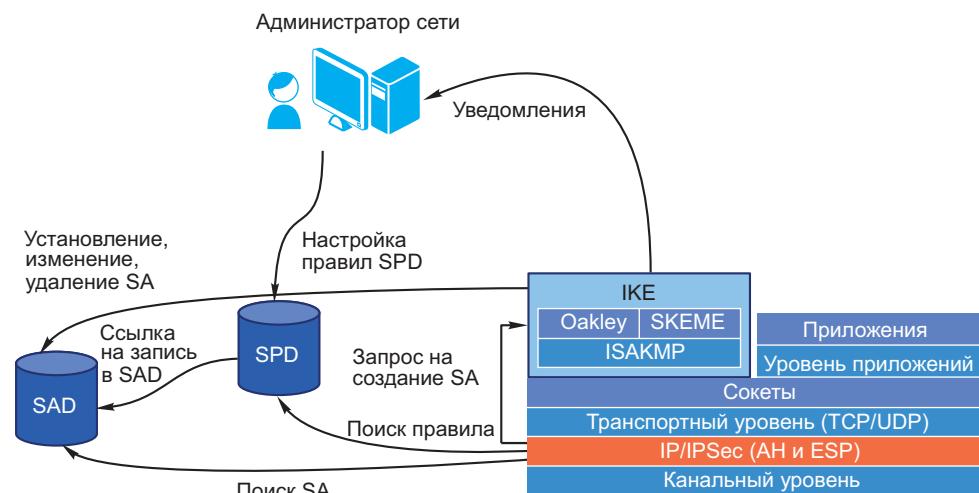


Рис. 3.56. Компоненты IPSec и их взаимодействие

Давайте подведем краткий итог и опишем процесс обработки IP-трафика. IPSec представляет собой набор открытых стандартов для обеспечения защищенных коммуникаций через IP-сети. Он полагается на два протокола безопасности трафика: AH и ESP. Управление параметрами, необходимыми для работы этих протоколов, выполняется безопасной ассоциацией (SA), т. е. ассоциацией, которая содержит параметры, используемые для защиты данной части трафика. Безопасные ассоциации хранятся в базе данных SAD. Они

устанавливаются, изменяются и удаляются с помощью протокола IKE. Защита, предлагаемая IPSec, основана на правилах, определенных в базе данных SPD. Для каждого пакета она позволяет определить, будут ли к нему применены какие-либо сервисы безопасности, будет ли он передан без защиты или будет отброшен.

При обработке исходящего трафика IPSec должен выполнить следующие шаги:

1. При поступлении пакета уровень IPSec обращается к базе SPD, чтобы определить, как его обрабатывать. Поля заголовка пакета проверяются на совпадение с селекторами, заданными в правилах, хранящихся в SPD.

2. Если произошло совпадение, пакет обрабатывается в соответствии с действием, указанным в правиле: отбрасывается, обходит IPSec или к нему применяется защита с помощью AH или ESP.

3. Если к пакету применяется защита, то запись SPD содержит ссылку на соответствующую запись в SAD, которая определяет режим, криптографические алгоритмы, ключи, SPI, PMTU и т. д. Если соответствующая SA существует, трафик аутентифицируется и/или шифруется.

4. Если вызываемая SA не существует, уровень IPSec инициирует работу протокола IKE для создания новой SA. Если SA успешно создана, в базе SPD создается новая запись для обработки исходящего трафика, в базе SAD создаются новые записи для исходящего и входящего трафика (для двухсторонней обработки создается пара SA). Пакет аутентифицируется и/или шифруется. Если SA не создана, пакет отбрасывается.

5. Пакет передается на сетевой интерфейс.

Обработка входящего трафика отличается от обработки исходящего.

1. Когда интерфейс получает пакет из сети, его обработка делится на две категории:

- если заголовок пакета защищен IPsec, предпринимается попытка найти соответствующую SA в базе SAD;

- если заголовок пакета не защищен IPsec, идет обращение к базе SPD с целью определения дальнейшей обработки пакета: отбросить его или пропустить.

2. Если пакет защищен протоколом AH или ESP, идет обращение к SAD с целью поиска соответствующей SA. Для поиска записи используется SPI или SPI + идентификатор протокола IPsec из заголовка пакета.

3. Если SA, соответствующая SPI (SPI + идентификатор протокола IPsec), не найдена в SAD, пакет отбрасывается и администратору отправляется соответствующее уведомление.

4. Если запись найдена, проверяется, соответствует ли пакет параметрам SA, через которую он был принят. Пакет аутентифицируется и/или расшифровывается. Если криптографическая обработка неуспешна, пакет отбрасывается и отправляется уведомление. В противном случае из пакета удаляются заголовки и концевики AH и ESP, и он передается вышестоящему уровню.

3.3.2. Протокол Encapsulating Security Payload (ESP)

В маршрутизаторах серии DIR-xxx и межсетевых экранах DFL-xxx по умолчанию поддерживается только протокол ESP. Поэтому здесь будет рассматриваться только он.

Протокол ESP описан в RFC 4303. Он может обеспечивать следующие сервисы безопасности:

- Конфиденциальность (необязательно);
- Целостность (обязательно);
- Конфиденциальность и целостность (обязательно).

Сервисы конфиденциальности и целостности могут использоваться в протоколе ESP независимо друг от друга. Однако так делать не рекомендуется. Использование только шифрования для обеспечения конфиденциальности данных обеспечивает защиту только от пассивных атак. Без использования целостности/аутентификации трафик становится уязвимым для некоторых видов активных атак. Сервис конфиденциальности может использоваться для повышения общей производительности системы в том случае, если вышеперечисленные уровни обеспечивают аутентификацию и целостность.

Сервисы аутентификации и целостности данных объединены в термине «целостность». Использование только сервиса целостности является альтернативой использования протокола AH, так как он быстрее в обработке и более приспособлен для встраивания во многие программные реализации. О применении только этого сервиса стороны должны договариваться в процессе работы протокола управления SA.

Несмотря на то что сервисы конфиденциальности и целостности могут использоваться независимо, стандарт рекомендует использовать их одновременно.

Формат пакета ESP

В пакете ESP можно выделить следующие части (рис. 3.57):

- Заголовок ESP (ESP Header);
- Нагрузка ESP (ESP Payload);
- Концевик ESP (ESP Tailer).

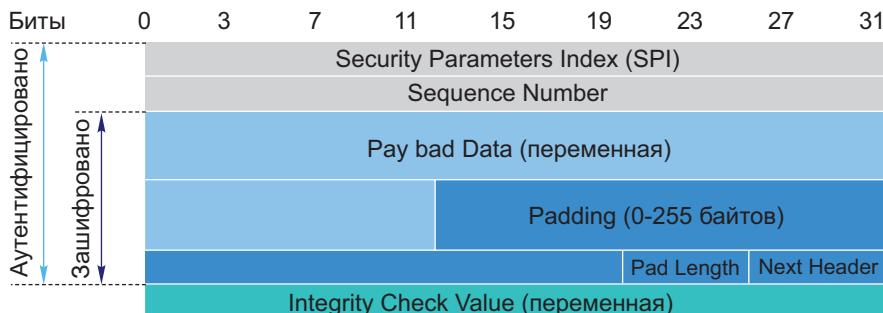


Рис. 3.57. Формат пакета ESP

3. Протокол IP

Заголовок пакета ESP (рис. 3.58) разработан для обеспечения комбинаций различных сервисов в IPv4 и IPv6. Он вставляется после IP-заголовка и перед заголовком протокола вышележащего уровня (транспортный режим) или перед IP-заголовком инкапсулированной дейтаграммы (туннельный режим). Поля *Протокол* (IPv4) или *Следующий заголовок* (IPv6) в заголовке внешнего по отношению к ESP протокола должны содержать значение 50.

Заголовок ESP содержит два поля: Security Parameters Index (SPI) и Sequence Number. Эти поля передаются в открытом виде.

Поле *Security Parameters Index* (SPI) — 32-битное значение, которое вместе с IP-адресом назначения и протоколом безопасности (в данном случае ESP) однозначно идентифицируют SA для данной дейтаграммы.

Поле *Sequence Number* — 32-битное значение монотонно возрастающего счетчика последовательности. Это значение используется для защиты от replay-атак. Обработка поля Sequence Number выполняется на стороне получателя; отправитель должен всегда передавать данное поле. Счетчик отправителя и счетчик получателя имеют значение 0 при установлении SA. Первый пакет, посылаемый с использованием данной SA, имеет Sequence Number, равный 1.

39 4.821177000 192.168.20.20	192.168.20.10	ESP	150 ESP (SPI=0x755a4912)
Frame 39: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits)			
Ethernet II, Src: a0:ab:1b:a4:95:53 (a0:ab:1b:a4:95:53), Dst: a0:ab:1b:a4:95:59 (a0:ab:1b:a4:95:59)			
Internet Protocol Version 4, Src: 192.168.20.20 (192.168.20.20), Dst: 192.168.20.10 (192.168.20.10)			
Version: 4 Header length: 20 bytes Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport)) Total Length: 136 Identification: 0xe00b (57355) Flags: 0x00 Fragment offset: 0 Time to live: 255 Protocol: ESP (50) Header checksum: 0x31c9 [correct] Source: 192.168.20.20 (192.168.20.20) Destination: 192.168.20.10 (192.168.20.10)			
Encapsulating Security Payload ESP SPI: 0x755a4912 ESP Sequence: 9			
0020 14 0a 76 5a 49 12 00 00 00 09 9a ec 38 f3 2d d1 ..JZ...8.-. 0030 77 7b b4 b7 44 2d 81 6b 8f 78 73 91 21 9e 1c 36 wI..D..k .xs.!..6 0040 dd 7d 1f 53 c3 fa 66 e1 a7 33 da 10 32 6c e6 c9 }S..f ..3..21.. 0050 d4 54 f6 4a 46 bb 0e e9 4e 31 71 49 86 f2 0b a3 .Z.JF... NiqI... 0060 30 1c 8b 9e 2d 77 f8 cc d9 56 a0 e1 70 02 d7 1b 0...-w...V..p... 0070 42 66 41 7c ba cc a1 47 b0 25 d6 b3 92 1f ab 46 Bfa]...G .%....F 0080 13 db 53 b3 cb 85 12 84 8f db 14 e0 95 c5 47 b8 ..S.....G. 0090 9e a4 f4 ba 52 89			

Рис. 3.58. Пакет ESP

Следующей частью пакета ESP является нагрузка. *Payload Data* является обязательным полем переменной длины, содержащим данные вышележащего протокола, тип которого указан в поле *Next Header*. Длина данного поля равна целому числу байтов. Если алгоритм, используемый для шифрования, требует криптографически синхронизованных данных, например вектор инициализации IV, то эти данные также содержатся в этом поле.

Третьей частью пакета ESP является концевик. Он включает поля *Padding*, *Pad Length*, *Next Header* и используется для шифрования.

Шифруются нагрузка и концевик ESP. Заголовок ESP передается в открытом виде.

Использование поля *Padding* (Добавление) обусловлено несколькими факторами.

- Если используемый алгоритм шифрования требует, чтобы незашифрованный текст был кратен определенному количеству байтов, например размеру блока для блочных алгоритмов, то поле *Padding* используется для дополнения незашифрованных данных (состоящих из полей *Payload Data*, *Pad Length* и *Next Header*) до размера, требуемого алгоритмом.

- Добавление может также требоваться независимо от алгоритма шифрования для гарантирования того, что полученные зашифрованные данные завершаются на 4-байтной границе. Поля *Pad Length* и *Next Header* должны быть, таким образом, расположены в 4-байтном слове, чтобы гарантировать, что поле *ICV* привязано к 4-байтной границе.

Добавление может быть использовано для маскирования реальной длины содержимого для обеспечения конфиденциальности потока трафика. Однако следует понимать, что использование такого добавления увеличивает трафик.

Поле *Pad Length* указывает число байтов добавления, которые расположены непосредственно перед этим полем. *Next Header* является 8-битовым полем, которое указывает тип данных, содержащихся в поле *Payload Data*, например, заголовок Extension в IPv6 или идентификатор протокола более высокого уровня. Значение данного поля выбирается из множества IP Protocol Number, определяемого IANA.

Опциональное поле *Integrity Check Value* (ICV) переменной длины завершает пакет. Оно вычисляется для полей заголовка, нагрузки и концевика ESP. Это поле добавляется после концевика в том случае, если алгоритм обеспечения целостности использует ICV.

Размещение заголовков

Протокол ESP может использоваться в двух режимах: транспортном и туннельном. Первый режим применяется в основном для создания VPN между двумя узлами и обеспечивает защиту для протоколов более высокого уровня, но не для заголовка IP. Заметим, что в данном режиме для реализаций «*bump-in-the-stack*» и «*bump-in-the-wire*» может потребоваться реассемблирование входящих и исходящих IP-фрагментов.

В транспортном режиме заголовок ESP расположен после IP-заголовка и перед заголовком протокола более высокого уровня, например, TCP, UDP, ICMP и т. д. (рис. 3.59). На рис. 3.60 показано добавление заголовка в транспортном режиме ESP для типичного пакета IPv4.

В протоколе IPv6 заголовок ESP рассматривается как расширенный, и таким образом он должен появляться после расширенных заголовков Hop-by-Hop, Routing и Fragmentation. Расширенные заголовки Destination Options могут размещаться как до ESP, так и после него.

3. Протокол IP

Туннельный режим ESP может быть реализован как на узлах, так и на шлюзах безопасности. При использовании ESP на шлюзе безопасности для защиты транзитного трафика должен применяться туннельный режим.

До применения ESP

Исходный заголовок IP + возможные опции	Заголовок TCP/UDP	Данные
---	-------------------	--------

После применения ESP



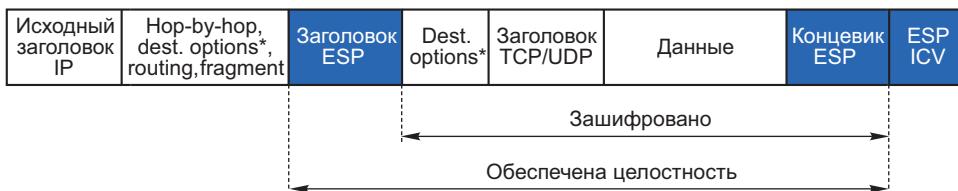
Рис. 3.59. Размещение заголовка ESP в транспортном режиме в пакете IPv4

В туннельном режиме внутренний IP-заголовок содержит конечные адреса источника и получателя, а внешний IP-заголовок содержит IP-адреса шлюзов безопасности. В туннельном режиме ESP защищает весь внутренний IP-пакет, включая весь внутренний IP-заголовок.

До применения ESP

Исходный заголовок IP	Расширенные заголовки (если есть)	Заголовок TCP/UDP	Данные
-----------------------	-----------------------------------	-------------------	--------

После применения ESP

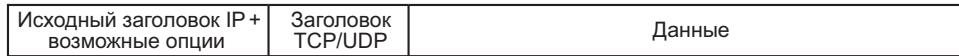


* = если присутствует, может быть до ESP, после ESP, или там и там

Рис. 3.60. Размещение заголовка ESP в транспортном режиме для пакета IPv6

Расположение ESP в туннельном режиме относительно внешнего IP-заголовка является тем же самым, что и для транспортного режима. Рис. 3.61 и рис. 3.62 иллюстрируют добавление заголовков в туннельном режиме ESP для типичных пакетов IPv4 и IPv6 соответственно.

До применения ESP



После применения ESP

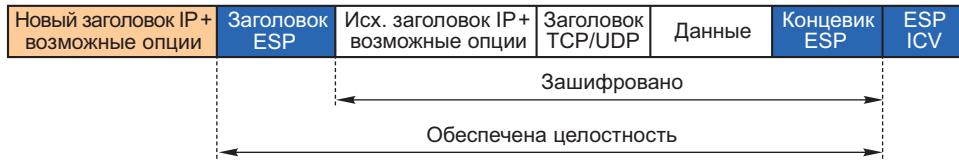


Рис. 3.61. Размещение заголовка ESP в туннельном режиме в пакете IPv4

До применения ESP



После применения ESP

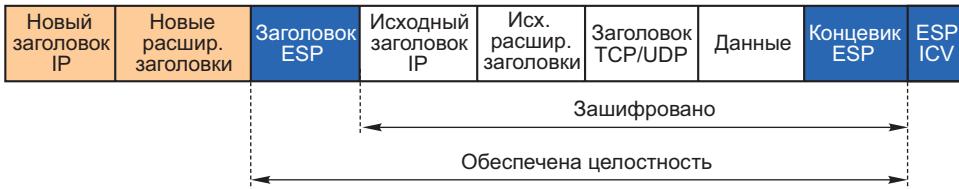


Рис. 3.62. Размещение заголовка ESP в туннельном режиме в пакете IPv6

Обработка исходящего пакета

В транспортном режиме отправитель инкапсулирует информацию протокола верхнего уровня в ESP-заголовок и сохраняет без изменения IP-заголовок (и любые IP-заголовки расширения в протоколе IPv6). В туннельном режиме внешний и внутренний IP-заголовки могут по-разному соотноситься друг с другом.

1. Поиск подходящей SA

ESP применяется к исходящему пакету после того, как определено, какой SA он принадлежит.

2. Шифрование пакета

Отправитель выполняет следующие действия:

- Инкапсуляция в поле *ESP Payload*:

- для транспортного режима — только исходная информация протокола верхнего уровня;
- для туннельного режима — вся исходная IP-дейтаграмма;

- Добавление необходимого дополнения (поле *Padding*);
- Шифрование полученных данных (*Payload Data*, *Padding*, *Pad Length*, *Next Header*), используя ключ, алгоритм шифрования, режим алгоритма, указанные в SA.

Первым выполняется шифрование, затем выполняется аутентификация (обеспечение целостности), поэтому поле *ICV* не зашифровано. Такой порядок обработки дает получателю возможность быстро обнаружить и отбросить повторные или фиктивные пакеты, что потенциально снижает вероятность успешного выполнения DoS-атак. Это также допускает возможность параллельной обработки пакетов получателем, например расшифрование может выполняться одновременно с проверкой целостности. Заметим, что поскольку *ICV* не защищено шифрованием, для его вычисления должен применяться алгоритм обеспечения целостности с ключом.

3. Создание Sequence Number

Первый пакет, посылаемый по вновь созданной SA, имеет *Sequence Number*, равный 1. Если используется защита от replay-атак, отправитель проверяет, что значение счетчика не переполнилось перед созданием нового значения в поле *Sequence Number*. Другими словами, отправитель не должен посылать пакет по SA, если возникает переполнение *Sequence Number*.

Отправитель снова посыпает пакет, если он не получил уведомления о его получении. Если счетчик переполнился, отправитель должен установить новую SA и вычислить новые ключи.

4. Вычисление значения проверки целостности

Если для SA требуется обеспечение целостности, отправитель вычисляет ICV для полей заголовка, нагрузки и концевика ESP. Таким образом, для полей *SPI*, *Sequence Number*, *Payload Data*, *Padding* (если присутствует) и *Next Header* вычисляется ICV. Заметим, что последние четыре поля являются зашифрованными, так как шифрование выполняется до вычисления ICV.

Для некоторых алгоритмов обеспечения целостности строка байтов, для которой вычисляется ICV, должна быть кратна длине блока выбранного алгоритма. Если длина данной строки байтов не соответствует требуемой длине блока, то до вычисления ICV должно выполняться добавление в конец ESP-пакета (после поля *Next Header*). Октеты добавления должны иметь нулевые значения. Длина блока и, следовательно, длина добавления определяются из спецификации алгоритма. Данное добавление не передается вместе с пакетом.

5. Фрагментация

После описанных выше ESP-преобразований при необходимости может выполняться фрагментация пакета. Транспортный режим ESP применяется только к целым IP-дейтаграммам (не к фрагментам IP). Далее IP-пакет, для которого применено ESP-преобразование, может быть фрагментирован маршрутизаторами. Фрагменты должны повторно собираться перед ESP-преобразованием на стороне получателя.

В туннельном режиме ESP применяется к IP-пакету, который может быть фрагментом IP-дейтаграммы.

Обработка входящего пакета

1. Реассемблирование

Если требуется реассемблирование, то оно выполняется до ESP-обработки.

2. Поиск подходящей SA

При получении пакета, содержащего ESP-заголовок, получатель определяет подходящую одностороннюю SA, просматривая SAD. Для односторонних SA это определяется на основе значения SPI в заголовке. В записи SAD указано, следует ли проверять последовательный номер. Также в записи SAD указаны алгоритмы и ключи, используемые для расшифровывания.

Если для данного пакета не существует SA, получатель отбрасывает пакет. Затем это событие записывается в лог с указанием SPI, даты и времени получения, адресов отправителя и получателя, последовательного номера и, возможно, другой информации.

Заметим, что трафик управления SA, такой как IKE-пакеты, не идентифицируются SPI.

3. Проверка номера последовательности

В основе защиты от replay-атак лежит проверка корректного значения номера последовательности. Получатель проверяет, что пакет содержит номер последовательности, который не является дубликатом номера последовательности другого пакета, полученного в данной SA. Такая проверка выполняется первой, после того как найдена соответствующая SA, чтобы как можно быстрее отбросить дублирующие пакеты.

3.3.3. Протокол Internet Key Exchange (IKE)

Протокол IPSec, как и многие другие протоколы сетевой безопасности, основан на концепции общего секрета (ключа). Два устройства, которые хотят безопасно обмениваться информацией, шифруют и расшифровывают ее с помощью секрета, известного только им. Третий, кто не владеет секретом, может перехватить информацию, но прочитать ее в большинстве случаев не сможет (возможность расшифровывания зависит от стойкости алгоритма и длины ключа). Поэтому, прежде чем AH или ESP начнет работу, необходимо, чтобы два взаимодействующих устройства обмениались секретом, который протоколы будут использовать.

Напомним, что IPSec предполагает возможность как автоматического создания SA, так и создания SA вручную. Распределение ключей вручную является простейшим способом создания SA, определения алгоритмов и ключей, при котором администратор вручную конфигурирует каждую систему, указывая ключи и другие параметры. Такая технология применяется в маленьких статичных окружениях и не масштабируется.

Для автоматического создания и управления SA используется протокол IKE. Первая версия IKE была определена в RFC 2407, 2408, 2409. Протокол IKE версии 2 определен в ряде RFC, начиная с RFC 4306. Протокол IKEv2 обратно *не совместим* с протоколом IKEv1, поэтому оборудование на разных концах линии связи должно поддерживать одинаковую версию протокола.

3.3.3.1. Протокол IKEv1

В IKEv1 для описания форматов передаваемых сообщений используется протокол ISAKMP. Он определяет форматы пакетов для ведения переговоров об установлении, изменении и удалении SA. Протокол IKE обеспечивает Perfect Forward Secrecy (PFS). Термин PFS (или его синоним Forward Secrecy) описывает свойство определенных протоколов обмена ключами, которое основано на следующей концепции: компрометация одного ключа позволит получить доступ к данным, защищенным только этим одним ключом. IKE реализован поверх протокола UDP. Порт UDP с номером 500 зарезервирован для IKE в IANA. Производители могут дополнительно поддерживать IKE поверх других транспортных протоколов или поверх IP.

В протоколе ISAKMP определены две фазы переговоров (рис. 3.63).

- Фаза I: во время первой фазы два участника ISAKMP договариваются о том, как защищать дальнейший трафик. В результате переговоров между двумя сторонами устанавливается безопасный аутентифицированный канал, называемый IKE Security Association (IKE SA). IKE SA является двунаправленной, т. е. создается сразу в двух направлениях.

- Фаза II: IKE SA, установленная в первой фазе, используется во второй фазе для безопасного обмена информацией при создании IPSec SA для протокола AH или ESP. При этом во второй фазе может быть установлено несколько безопасных ассоциаций.

Все взаимодействия IKE состоят из пары сообщений: запрос и ответ. Эта пара называется «обмен». В каждой из фаз выполняется несколько обменов сообщениями. В фазе I обмены происходят в одном из двух режимов: Main Mode или Aggressive Mode. В фазе II обмен выполняется в единственном режиме Quick Mode.

Обмен фазы I

Участники фазы I договариваются о следующих обязательных параметрах:

- алгоритме шифрования;
- алгоритме хеширования;
- методе аутентификации;
- информации о группе для алгоритма Диффи — Хеллмана.

Кроме того, возможны дополнительные переговоры о псевдослучайной функции (PRF). Если переговоры о ней не ведутся, то используется HMAC с хеш-алгоритмом, о котором договорились (MD5, SHA). Группа Диффи — Хеллмана определяется по номеру.

В *главном режиме* (Main Mode) переговоры об установлении IKE SA выполняются путем обмена тремя парами сообщений ISAKMP (т. е. между

узлами передается шесть сообщений). Сообщение ISAKMP инкапсулировано в дейтаграмму UDP с номерами порта назначения и источника, установленными в 500. Сообщение ISAKMP имеет заголовок ISAKMP и одну или более нагрузок (payload), как определено в RFC 2408.

Узел, который начинает обмен, называется инициатором (initiator), узел, отвечающий на сообщение инициатора, — ответчик (responder).

В первой паре сообщений (первое (рис. 3.64) и второе (рис. 3.64) сообщения) конечные точки договариваются об используемых алгоритмах шифрования, аутентификации, хеширования, группах Диффи — Хеллмана. Они конфигурируются администратором на устройстве.

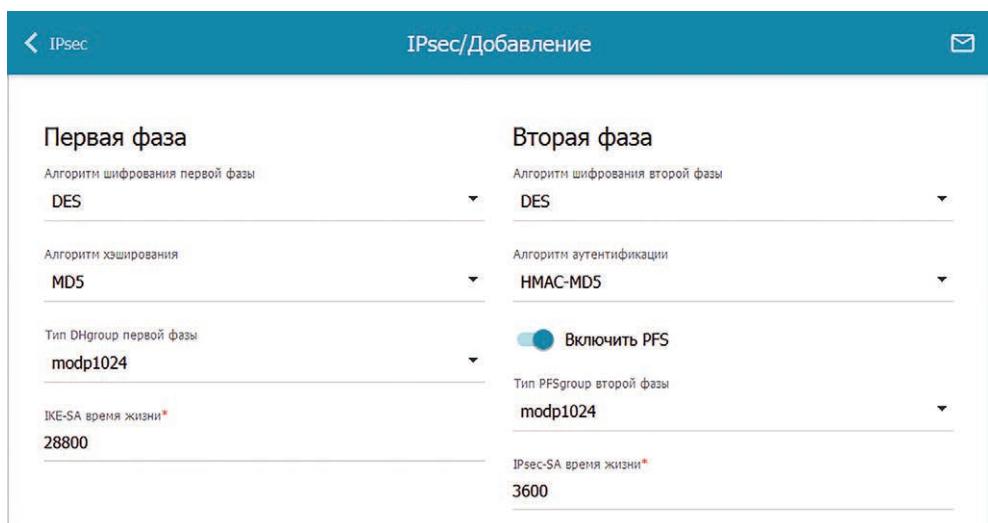


Рис. 3.63. Пример настройки параметров первой и второй фазы IKE на маршрутизаторе D-Link

Алгоритм распределения ключей Диффи — Хеллмана используется для безопасной генерации общего секретного ключа для конечных точек, между которыми отсутствует защищенное соединение. Далее этот общий секрет используется для генерации значения, используемого для вычисления секретных ключей первой и второй фазы. Группа Диффи — Хеллмана определяется по номеру. Номер каждой группы соответствует длине ключа и типу криптографического генератора.

Помимо согласования параметров, первая пара сообщений также включает обмен «cookie». Их использование может препятствовать некоторым попыткам DoS-атак, состоящим в простом наводнении пакетами (flooding-атаки). Абсолютная защита от DoS-атак невозможна, но такие «cookie» обеспечивают возможность более быстрого ее определения.

3. Протокол IP

```
☒ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
☒ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 0000000000000000
    Next payload: Security Association (1)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
☒ Flags: 0x00
    Message ID: 0x00000000
    Length: 160
☒ Type Payload: Security Association (1)
    Next payload: Vendor ID (13)
    Payload length: 52
    Domain of interpretation: IPSEC (1)
☒ Situation: 00000001
☒ Type Payload: Proposal (2) # 1
    Next payload: NONE / No Next Payload (0)
    Payload length: 40
    Proposal number: 1
    Protocol ID: ISAKMP (1)
    SPI Size: 0
    Proposal transforms: 1
☒ Type Payload: Transform (3) # 1
    Next payload: NONE / No Next Payload (0)
    Payload length: 32
    Transform number: 1
    Transform ID: KEY_TKE (1)
    ☒ Transform IKE Attribute Type (t=11, l=2) Life-Type : Seconds
    ☒ Transform IKE Attribute Type (t=12, l=2) Life-Duration : 28800
    ☒ Transform IKE Attribute Type (t=1, l=2) Encryption-Algorithm : DES-CBC
    ☒ Transform IKE Attribute Type (t=3, l=2) Authentication-Method : PSK
    ☒ Transform IKE Attribute Type (t=2, l=2) Hash-Algorithm : SHA
    ☒ Transform IKE Attribute Type (t=4, l=2) Group-Description : Alternate 1024-bit MODP group
☒ Type Payload: Vendor ID (13) : Unknown Vendor ID
☒ Type Payload: Vendor ID (13) : draft-ietf-ipsec-nat-t-ike-02\n
☒ Type Payload: Vendor ID (13) : RFC 3947 Negotiation of NAT-Traversal in the IKE
☒ Type Payload: Vendor ID (13) : RFC 3706 DPD (Dead Peer Detection)
```

Рис. 3.64. Первое сообщение Main mode

Вторая пара сообщений (третье (рис. 3.65) и четвертое сообщения) выполняет обмен Диффи — Хеллмана, используя параметры, согласованные в результате обмена первой парой сообщений. Инициатор и ответчик обмениваются открытыми значениями Диффи — Хеллмана и случайными значениями (nonce), необходимыми для обмена. Метод аутентификации, определенный при переговорах в первых сообщениях, влияет на содержимое сообщений второй пары. Всего определено четыре метода аутентификации: цифровая подпись, две формы аутентификации с открытым ключом, аутентификация на основе PSK. Сообщения, включающие аутентификацию на основе PSK или цифровой подписи, имеют аналогичные поля — *Заголовок*, *KE* (Key Exchange) и *nonce*. Сообщения, включающие аутентификацию с шифрованием открытым ключом, имеют поля *Заголовок*, *KE* (Key Exchange), зашифрованное *nonce* и идентификаторы обмена, защищенные общими ключами.

Последние два сообщения (пятое (рис. 3.65) и шестое сообщения) аутентифицируют обмен Диффи — Хеллмана. Другими словами, стороны обмена аутентифицируют друг друга. Как происходит эта аутентификация, зависит от ранее согласованного метода. При этом, несмотря на используемый метод

Технологии TCP/IP в современных компьютерных сетях

```
☒ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
☒ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bfff19
    Next payload: Security Association (1)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
☒ Flags: 0x00
    Message ID: 0x00000000
    Length: 140
☒ Type Payload: Security Association (1)
    Next payload: Vendor ID (13)
    Payload length: 52
    Domain of interpretation: IPSEC (1)
☒ Situation: 00000001
☒ Type Payload: Proposal (2) # 1
    Next payload: NONE / No Next Payload (0)
    Payload length: 40
    Proposal number: 1
    Protocol ID: ISAKMP (1)
    SPI Size: 0
    Proposal transforms: 1
    Type Payload: Transform (3) # 1
        Next payload: NONE / No Next Payload (0)
        Payload length: 32
        Transform number: 1
        Transform ID: KEY_IKE (1)
    ☒ Transform IKE Attribute Type (t=11, l=2) Life-Type : Seconds
    ☒ Transform IKE Attribute Type (t=12, l=2) Life-Duration : 28800
    ☒ Transform IKE Attribute Type (t=1, l=2) Encryption-Algorithm : DES-CBC
    ☒ Transform IKE Attribute Type (t=3, l=2) Authentication-Method : PSK
    ☒ Transform IKE Attribute Type (t=2, l=2) Hash-Algorithm : SHA
    ☒ Transform IKE Attribute Type (t=4, l=2) Group-Description : Alternate 1024-bit MODP group
☒ Type Payload: Vendor ID (13) : RFC 3947 Negotiation of NAT-Traversal in the IKE
    Next payload: Vendor ID (13)
    Payload length: 20
    Vendor ID: 4a131c81070358455c5728f20e95452f
    Vendor ID: RFC 3947 Negotiation of NAT-Traversal in the IKE
☒ Type Payload: Vendor ID (13) : RFC 3706 DPD (Dead Peer Detection)
    Next payload: Vendor ID (13)
    Payload length: 20
    Vendor ID: afcad71368a1f1c96b8696fc77570100
    Vendor ID: RFC 3706 DPD (Dead Peer Detection)
☒ Type Payload: Vendor ID (13) : KAME/racoon
    Next payload: NONE / No Next Payload (0)
    Payload length: 20
    Vendor ID: 7003cbc1097dbe9c2600ba6983bc8b35
    Vendor ID: KAME/racoon
```

Рис. 3.65. Второе сообщение Main mode

аутентификации, нагрузка третьей пары сообщений шифруется с помощью информации, полученной в результате обмена второй парой сообщений.

Подведем краткий итог: главный режим использует три пары сообщений. Каждая из этих пар сообщений имеет различное назначение. Первая пара сообщений согласует параметры IKE SA, вторая пара выполняет обмен ключами, третья пара взаимно аутентифицирует конечные точки соединения.

Агрессивный режим (Aggressive Mode) предлагает быструю альтернативу главному режиму. Стороны договариваются об установлении IKE SA путем обмена тремя сообщениями, а не тремя парами сообщений. В первых двух сообщениях конечные точки договариваются об используемых алгоритмах, обмениваются открытыми значениями Диффи — Хеллмана, случайными

3. Протокол IP

```
■ Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
■ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
■ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bffc19
    Next payload: Key Exchange (4)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
    Flags: 0x00
    Message ID: 0x00000000
    Length: 248
    ■ Type Payload: Key Exchange (4)
        Next payload: Nonce (10)
        Payload length: 132
        Key Exchange Data: f2a20d5fe89ddc4af1d29b0b588167750f06db104eef4efc...
    ■ Type Payload: Nonce (10)
        Next payload: Vendor ID (13)
        Payload length: 20
        Nonce DATA: d436e6532b7aab52e6203957b71a8f18
    ■ Type Payload: Vendor ID (13) : KAME/racoon
    ■ Type Payload: NAT-D (RFC 3947) (20)
    ■ Type Payload: NAT-D (RFC 3947) (20)
```

Рис. 3.66. Третье сообщение Main mode (аутентификация на основе PSK)

```
■ Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
■ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
■ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bffc19
    Next payload: Identification (5)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
    Flags: 0x01
    Message ID: 0x00000000
    Length: 68
    Encrypted Data (40 bytes)
```

Рис. 3.67. Пятое сообщение Main mode (аутентификация на основе PSK)

значениями (nonce) и идентификациями. Второе и третье сообщения взаимно аутентифицируют инициатора и ответчика.

В агрессивном режиме согласуются те же самые параметры, что и в главном режиме, определены те же методы аутентификации. Однако, несмотря на быстроту, агрессивный режим менее гибкий и безопасный. Поскольку обмен ключами по алгоритму Диффи — Хеллмана начинается с первого пакета, стороны не имеют возможности договориться о параметрах Диффи — Хеллмана. Обе стороны обмениваются информацией до того, как создана защищенная IKE SA. Таким образом, информация, подтверждающая идентичность в агрессивном режиме, не всегда скрыта. Если злоумышленник подключится к линии, он сможет определить стороны, выполняющие переговоры.

Обмен фазы II

Основная цель фазы II установить IPSec SA для фактического IPSec-соединения. В отличие от IKE SA эта SA является односторонней. Это означает, что IPSec-соединение между двумя устройствами требует двух SA (по одной в каждом направлении). Пара SA (входящая и исходящая) создается путем обмена тремя сообщениями в *быстром режиме* (Quick Mode).

Сам по себе быстрый режим не является законченным обменом. Он связан с фазой I, так как использует ключи, вычисленные в процессе ее выполнения. Все содержимое сообщений, которыми обмениваются в быстром режиме, за исключением заголовка ISAKMP, будет зашифровано алгоритмами и ключами, определенными в фазе I.

Quick Mode выполняет переговоры об IPSec SA и обменивается случайными значениями (nonce), которые обеспечивают защиту от повторов.Nonce используются для создания нового материала ключа и предотвращения replay-атак, в результате которых могут быть созданы ложные SA.

В первом сообщении инициатор обмена отправляет ключевой материал, nonce, предлагаемые параметры SA и HASH для аутентификации (рис. 3.68). По умолчанию Quick Mode вычисляет ключи из ключевого материала, созданного в фазе I. Если при настройке фазы II на устройстве была активизирована функция Perfect Forward Secrecy (PFS), то для каждой SA, созданной на фазе II, будут полностью обновлены ключи путем дополнительного обмена Диффи — Хеллмана. Несмотря на то что PFS увеличивает издержки по установлению SA, она обеспечивает большую защиту, так как ни один из ключей не связан с другим. Если какой-либо ключ будет скомпрометирован, атакующий не сможет получить доступ к другим IPSec SA.

```
④ Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
④ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
④ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bfffcc19
    Next payload: Hash (8)
    Version: 1.0
    Exchange type: Quick Mode (32)
    Flags: 0x01
        .... .1 = Encryption: Encrypted
        .... .0. = Commit: No commit
        .... .0.. = Authentication: No authentication
    Message ID: 0x94aeb81b
    Length: 156
    Encrypted Data (128 bytes)
```

Рис. 3.68. Первое сообщение Quick Mode

Во втором сообщении ответчик отправляет ключевой материал, nonce, выбранные параметры SA и HASH для аутентификации. В третьем сообщении инициатор отправляет HASH для аутентификации.

Для каждой SA (одной входящей и одной исходящей) назначаются разные SPI (один SPI выбирает инициатор, другой — ответчик). Это гарантирует создание различных ключей для каждого направления.

Как только ответчик подтверждает третье сообщение, SA устанавливаются и в SAD создаются соответствующие записи.

Обе IKE SA и IPSec SA обычно имеют ограниченное время жизни (*lifetime*). Если время жизни любой из SA подходит к концу, стороны должны создать новую SA через процесс смены ключей и использовать ее вместо предыдущей. Время жизни каждой SA можно задавать в настройках устройства.

Информационный обмен

В различные моменты времени стороны могут захотеть сообщить друг другу о возникших ошибках или уведомить о возникших событиях. Для выполнения этого определен информационный обмен IKE (рис. 3.69). Информационные обмены должны осуществляться только после выполнения обмена ключом, чтобы они были криптографически защищены. Это гарантирует, что неавторизованные сообщения не нарушают переговоры IPsec или преждевременно не завершат IPsec SA.

```

■ Internet Protocol Version 4, Src: 192.168.20.20 (192.168.20.20), Dst: 192.168.20.10 (192.168.20.10)
■ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
■ Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bffc19
    Next payload: Hash (8)
    Version: 1.0
    Exchange type: Informational (5)
    Flags: 0x01
        .... .1 = Encryption: Encrypted
        .... .0. = Commit: No commit
        .... .0.. = Authentication: No authentication
    Message ID: 0xea6aff3c
    Length: 84
    Encrypted Data (56 bytes)

```

Рис. 3.69. Сообщение информационного обмена

С помощью информационного обмена одна конечная точка может, например, сообщить другой, что определенная SA больше не будет использоваться. Так как сообщения информационного обмена помещаются в дейтаграммы UDP и получатель не подтверждает их, нет никаких гарантий, что другая конечная точка их получит.

3.3.3.2. Протокол IKEv2

Протокол IKEv2 был первоначально описан в RFC 4306. Сейчас этот RFC считается устаревшим, актуальным является RFC 7296.

IKEv1 и IKEv2 обратно не совместимы, поскольку в IKEv2 изменен порядок обмена. В протоколе IKEv1 определен обмен фазы I и обмен фазы II. В фазе I стороны обмениваются шестью или тремя сообщениями в зависимости от используемого режима. В фазе II стороны обмениваются тремя сообщениями.

Взаимодействие по протоколу IKEv2 начинается с двух обменов: IKE_SA_INIT (рис. 3.70) и IKE_AUTH (рис. 3.71). Этот первоначальный обмен обычно состоит из четырех сообщений, хотя в некоторых случаях их количество может увеличиться. Это зависит от сложности выполняемой аутентификации.

Первая пара сообщений (IKE_SA_INIT) является начальным обменом, в котором узлы устанавливают IKE SA. С их помощью согласовываются криптографические алгоритмы, отправляются nonce и выполняется обмен Диффи — Хеллмана.

Технологии TCP/IP в современных компьютерных сетях

```
Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Internet Security Association and Key Management Protocol
    Initiator cookie: cc07c11990512a37
    Responder cookie: 0000000000000000
    Next payload: Security Association (33)
    Version: 2.0
    Exchange type: IKE_SA_INIT (34)
    Flags: 0x08
    Message ID: 0x00000000
    Length: 356
    Type Payload: Security Association (33)
    Type Payload: Key Exchange (34)
    Type Payload: Nonce (40)
    Type Payload: Notify (41)
    Type Payload: Notify (41)
    Type Payload: Vendor ID (43) : Unknown Vendor ID
    Type Payload: Vendor ID (43) : RFC 3947 Negotiation of NAT-Traversal in the IKE
```

Рис. 3.70. Первое сообщение обмена IKE_SA_INIT

Вторая пара сообщений (IKE_AUTH) аутентифицирует предыдущие сообщения, обменивается идентичностями и сертификатами, устанавливает первую и обычно единственную Child SA для протокола AH или ESP. Содержимое этих сообщений зашифровано, и целостность их защищена с помощью ключей, установленных в процессе обмена IKE_SA_INIT. Таким образом, идентичности скрыты от перехватчиков и все поля сообщений аутентифицированы.

```
Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Internet Security Association and Key Management Protocol
    Initiator cookie: cc07c11990512a37
    Responder cookie: 52a8ad6d56ddc6c1
    Next payload: Encrypted and Authenticated (46)
    Version: 2.0
    Exchange type: IKE_AUTH (35)
    Flags: 0x08
    Message ID: 0x00000001
    Length: 220
    Type Payload: Encrypted and Authenticated (46)
```

Рис. 3.71. Первое сообщение обмена IKE_AUTH

Все сообщения, следующие за первоначальным обменом, криптографически защищены с помощью криптографических алгоритмов и ключей, согласованных в процессе обмена IKE_SA_INIT.

Если дополнительно требуется создать новые Child SA или обновить ключи текущих IKE SA и Child SA, используется обмен CREATE_CHILD_SA. Этот обмен состоит из единственной пары сообщений.

Информационный обмен в IKEv2 выполняет те же функции, что и в IKEv1. Он позволяет сторонам доставлять друг другу управляющие сообщения об ошибках или уведомлять об определенных событиях. Информационные обмены должны осуществляться только после выполнения обмена ключом, чтобы они были криптографически защищены.

3.3.4. Использование NAT в протоколе IPSec

Для обеспечения целостности данных (одного из сервисов IPSec) запрещается какое-либо их изменение в процессе передачи. Это является основным препятствием, с которым можно столкнуться при совместном использовании NAT и IPSec. Все несовместимости между NAT и IPSec описаны в RFC 3715 «IPsec-Network Address Translation (NAT) Compatibility Requirements».

Поскольку NAT изменяет заголовок IP, это влияет на проверку целостности пакета IP в случае использования протокола АН. При любом режиме (транспортном или туннельном) протокол АН осуществляет аутентификацию отдельных частей заголовка IP. Именно поэтому протокол АН и NAT совместно работать не могут. Следовательно, единственный вариант — использовать протокол ESP в транспортном или в туннельном режиме. В туннельном режиме ESP защищает весь внутренний IP-пакет, включая весь внутренний IP-заголовок и заголовок транспортного уровня. Внешний заголовок IP остается нетронутым и может быть изменен, что и происходит при преобразовании NAT. Таким образом, этот процесс может функционировать до тех пор, пока речь не заходит о пакете TCP.

Однако как только возникает необходимость в передаче пакета TCP с преобразованием портов (NAPT), маршрутизатор NAT начинает работать с портами TCP и заново рассчитывает контрольную сумму заголовка TCP. А поскольку заголовок TCP зашифрован протоколом ESP, можно ожидать следующих проблем:

- Если заголовок TCP и данные зашифрованы, то маршрутизатор NAT не сумеет заново рассчитать контрольную сумму TCP, и передача данных окажется невозможна.
- Если заголовок TCP и данные не зашифрованы, то маршрутизатор NAT заново рассчитает контрольную сумму TCP и нарушит целостность данных с точки зрения протокола ESP.

Отсюда следует, что NAT и ESP способны совместно функционировать в транспортном режиме только в том случае, если сторонами IPSec отменена проверка контрольной суммы.

Но существует еще одна трудность: при использовании IKE с аутентификацией на основе PSK в большинстве реализаций IPSec происходит идентификация IP-адреса партнера по заранее заданному паролю. Изменение этого IP-адреса при использовании NAT может привести к сложностям с аутентификацией. Однако если идентификация партнера IPSec происходит на основе идентификационных данных (ID) пользователя или имени домена, то такая проблема не возникает.

Для решения вышеописанных проблем была создана функция NAT Traversal (NAT-T), которая позволяет передавать пакеты IPSec через NAT или NAPT, дополнительно упаковывая их в дейтаграмму UDP.

Работа протокола IKEv1 при использовании NAT описана в RFC 3947 «Negotiation of NAT-Traversal in the IKE».

Определение поддержки двумя сторонами IKEv1 функции NAT-Traversal и обнаружение устройств NAT на пути между ними выполняется на фазе I в обоих режимах.

RFC 3947 требует, чтобы в первых двух сообщениях фазы I стороны обменялись Vendor ID payload (рис. 3.72). Она является одной из нагрузок сообщений ISAKMP и используется для уведомления о поддержке функции NAT-Traversal. Содержимым Vendor ID payload является хеш-код MD5, вычисленный для строки «RFC 3947». После этого могут быть определены наличие NAT между двумя противоположными сторонами IKE и его расположение.

```
Frame 16: 202 bytes on wire (1616 bits), 202 bytes captured (1616 bits)
Ethernet II, Src: a0:ab:1b:f0:ef:b9 (a0:ab:1b:f0:ef:b9), Dst: a0:ab:1b:f0:ef:c8 (a0:ab:1b:f0:ef:c8)
Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 0000000000000000
    Next payload: Security Association (1)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
    Flags: 0x00
    Message ID: 0x00000000
    Length: 160
    Type Payload: Security Association (1)
    Type Payload: Vendor ID (13) : Unknown Vendor ID
    Type Payload: Vendor ID (13) : draft-ietf-ipsec-nat-t-ike-02\n
    Type Payload: Vendor ID (13) : RFC 3947 Negotiation of NAT-Traversal in the IKE
        Next payload: Vendor ID (13)
        Payload length: 20
        Vendor ID: 4a131c81070358455c5728f20e95452f
        Vendor ID: RFC 3947 Negotiation of NAT-Traversal in the IKE
    Type Payload: Vendor ID (13) : RFC 3706 DPD (Dead Peer Detection)
```

Рис. 3.72. Строки Vendor ID

Для определения NAT между двумя узлами необходимо определить, изменились ли IP-адреса и порты при пересылке. Для этого стороны отправляют друг другу нагрузки NAT Discovery (NAT-D), представленные на рис. 3.73. Их содержимым являются хеши IP-адресов и портов из пакетов обеих сторон. Нагрузки NAT-D включаются в третий и четвертый пакеты Main Mode и второй и третий пакеты Aggressive Mode.

Каждая сторона IKE отправляет две или более нагрузки NAT-D. Каждая нагрузка содержит один хеш. IP-адрес и номер порта назначения исходящего пакета IKE используются для вычисления хеша, который содержится в первой нагрузке NAT-D. IP-адрес и номер порта источника исходящего пакета IKE используются для вычисления хеша, который содержится во второй нагрузке NAT-D. Обычно стороны обмениваются только двумя нагрузками NAT-D. Однако если отправитель пакета имеет множество IP-адресов и не знает, какой адрес использовать для отправки пакета, он может поместить в пакет несколько нагрузок NAT-D для каждого IP-адреса.

Если на обоих концах вычислены те же самые хеши, что и в полученных сообщениях, это означает, что между участниками нет NAT. Если хеши не

3. Протокол IP

```
Frame 20: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
Ethernet II, Src: a0:ab:1b:f0:ef:b9 (a0:ab:1b:f0:ef:b9), Dst: a0:ab:1b:f0:ef:c8 (a0:ab:1b:f0:ef:c8)
Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Internet Security Association and Key Management Protocol
    Initiator cookie: 603761625616bf9c
    Responder cookie: 130f875f70bffc19
    Next payload: Key Exchange (4)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
Flags: 0x00
Message ID: 0x00000000
Length: 248
Type Payload: Key Exchange (4)
Type Payload: Nonce (10)
Type Payload: Vendor ID (13) : KAME/racoon
Type Payload: NAT-D (RFC 3947) (20)
    Next payload: NAT-D (RFC 3947) (20)
    Payload length: 24
    HASH of the address and port: a2136ce34fa7dfdea72f48f864c58383d391c476
Type Payload: NAT-D (RFC 3947) (20)
    Next payload: NONE / No Next Payload (0)
    Payload length: 24
    HASH of the address and port: ecc2ed9db8435c592724bc3a57165e861446923f
```

Рис. 3.73. Нагрузки NAT-D

совпадают, где-то была выполнена трансляция адреса или порта. Это означает, что для получения IPSec-пакетов необходимо обеспечить прохождение NAT.

Когда между двумя сторонами IKE обнаружено одно или более устройств NAT, в первом и втором сообщениях Quick mode не должно предлагаться использование транспортного или туннельного режима инкапсуляции. Для переговоров о прохождении NAT добавлены два новых режима инкапсуляции: UDP-Encapsulated-Transport и UDP-Encapsulated-Tunnel. Когда между двумя сторонами IKE нет NAT, переговоры об использовании этих режимов вестись не должны.

Для возможности изменения контрольных сумм TCP обоим участникам может понадобиться знать исходные IP-адреса, которые они использовали при создании пакета. Для этих целей RFC 3947 определил нагрузку NAT Original Address (NAT-OA), которая содержит IP-адреса.

Когда инициатор обмена в Quick mode отправляет предложение об использовании транспортного режима UDP-Encapsulated-Transport, в первом сообщении должны содержаться две нагрузки NAT-OA. Первая нагрузка NAT-OA содержит исходный IP-адрес инициатора, вторая — исходный IP-адрес ответчика. Когда ответчик принимает предложение об использовании режима UDP-Encapsulated-Transport, он отправляет две нагрузки NAT-OA во втором сообщении, которые содержат исходные IP-адреса инициатора и ответчика.

В туннельном режиме UDP-Encapsulated-Tunnel обе стороны не должны посыпать друг другу исходные IP-адреса.

RFC 7296 описывает механизм обнаружения NAT и определения его размещения на пути между двумя сторонами IKEv2. Для этого инициатор

и ответчик должны включать в свои сообщения IKE_SA_INIT нагрузку Notify типа NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP (рис. 3.74). Эти нагрузки могут использоваться для определения расположения NAT и стороны, расположенной за NAT.

```
Internet Protocol Version 4, Src: 192.168.20.10 (192.168.20.10), Dst: 192.168.20.20 (192.168.20.20)
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Internet Security Association and Key Management Protocol
    Initiator cookie: c007c11990512a37
    Responder cookie: 0000000000000000
    Next payload: Security Association (33)
    Version: 2.0
    Exchange type: IKE_SA_INIT (34)
    Flags: 0x08
    Message ID: 0x00000000
    Length: 356
    Type Payload: Security Association (33)
    Type Payload: Key Exchange (34)
    Type Payload:Nonce (40)
    Type Payload:Notify (41)
        Next payload: Notify (41)
        0... .... = Critical Bit: Not Critical
        Payload length: 28
        Protocol ID: RESERVED (0)
        SPI Size: 0
        Notify Message Type: NAT_DETECTION_SOURCE_IP (16388)
        Notification DATA: a05d259e6e6e44d000c8d234b0e24d93d10bf08b
    Type Payload:Notify (41)
        Next payload: Vendor ID (43)
        0... .... = Critical Bit: Not Critical
        Payload length: 28
        Protocol ID: RESERVED (0)
        SPI Size: 0
        Notify Message Type: NAT_DETECTION_DESTINATION_IP (16389)
        Notification DATA: 8a869b88feefadcc45f364a84494e8fd377d2669
    Type Payload: Vendor ID (43) : Unknown Vendor ID
    Type Payload: Vendor ID (43) : RFC 3947 Negotiation of NAT-Traversal in the IKE
```

Рис. 3.74. Нагрузки Notify

Каждая нагрузка содержит один хеш. SPI, IP-адрес и номер порта источника исходящего пакета IKE_SA_INIT используются для вычисления хеша, который содержится в нагрузке NAT_DETECTION_SOURCE_IP. Этих нагрузок в сообщении может быть много, если отправитель не знает, в какую из присоединенных сетей будет отправлен пакет.

SPI, IP-адрес и номер порта назначения исходящего пакета IKE_SA_INIT используются для вычисления хеша, который содержится в нагрузке NAT_DETECTION_DESTINATION_IP.

Если получатель определил, что полученные и вычисленные хеши NAT_DETECTION_SOURCE_IP и NAT_DETECTION_DESTINATION_IP не совпадают, он должен активизировать функцию NAT-Traversal. В случае, если не совпадают вычисленный и принятый хеш NAT_DETECTION_DESTINATION_IP, это означает, что система, получившая эту нагрузку, находится за NAT.

Если ни одна из полученных нагрузок NAT_DETECTION_SOURCE_IP не совпадает с ожидаемыми значениями IP-адреса и порта источника из

IP-заголовка, это означает, что система, отправившая эти нагрузки, находится за NAT.

Если между двумя сторонами IKEv2 определен NAT и при установлении Child SA ведутся переговоры об использовании транспортного режима, для защиты передаваемого по ней трафика будет использоваться режим UDP-Encapsulated-Transport. Если при установлении Child SA ведутся переговоры об использовании туннельного режима, для защиты передаваемого трафика будет использоваться режим UDP-Encapsulated-Tunnel. При этом стек TCP/IP должен обрабатывать полученные пакеты ESP, инкапсулированные в UDP, даже в том случае, если NAT не определен.

Для возможности изменения контрольных сумм TCP при использовании транспортного режима обоим участникам может понадобиться знать их исходные IP-адреса. Исходные IP-адреса содержатся в нагрузках Traffic Selector. Когда NAT был определен и инициатор предлагает использовать транспортный режим, запрос должен включать нагрузки Traffic Selector, которые содержат исходные IP-адреса инициатора и ответчика. Когда ответчик принимает предложение об использовании транспортного режима, его ответ содержит нагрузки Traffic Selector, которые содержат исходные IP-адреса инициатора и ответчика.

Способ передачи ESP-пакетов при обнаружении NAT описан в RFC 3948 «UDP Encapsulation of IPsec ESP Packets». Пакет IP, защищенный ESP, дополнительно упаковывается в дейтаграмму UDP, в результате чего он избегает проверки идентичности и целостности заголовка IP и TCP. Таким образом NAPT трактует ESP-пакет как обычную дейтаграмму UDP и обрабатывает ее обычным образом. Заголовок UDP вставляется перед заголовком ESP во всех режимах. Номера порта источника и порта приемника в заголовке UDP аналогичны номерам, используемым для пакетов IKE. Контрольная сумма всегда установлена в 0, что позволяет избежать проверки контрольной суммы промежуточными устройствами.

Использование порта UDP 4500

Преобразования NAT, которые знают о наличии IPSec, могут вызвать проблемы. Наилучшим способом избежать проблем является использование номера порта UDP, отличного от 500. Напомним, что UDP-порт 500 зарезервирован для IKE. RFC 3947 и RFC 7296 требуют, чтобы инициатор, расположенный за NAT, изменял номер порта UDP на 4500 сразу, как только определит существование NAT. Порт 4500 зарезервирован для трафика ESP и IKE, инкапсулированного в UDP.

В Main mode IKEv1 инициатор определяет существование NAT, когда обрабатывает четвертое сообщение и переключается на использование UDP-портов источника и назначения с номером 4500. В Aggressive mode IKEv1 инициатор определяет существование NAT, когда обрабатывает второе сообщение и переключается на использование UDP-портов источника и назначения с номером 4500 при отправке третьего сообщения.

В IKEv2 инициатор определяет существование NAT, когда обрабатывает ответ IKE_SA_INIT. Когда ответчик отправляет инициатору сообщение, он должен использовать значения портов из последнего сообщения, полученного от инициатора. После того как стороны определили наличие NAT, они должны отправлять все последующие сообщения с номером UDP-порта 4500. При этом RFC 7296 позволяет инициатору использовать UDP-порт 4500 для трафика IKEv2, независимо от того, существует NAT или нет, даже при отправке первого сообщения IKE_SA_INIT.

Инициатор всегда ожидает, что сообщения будут передаваться с порта 4500 и приниматься на порт 4500. Для ответчика, если он расположен за NAPT, номер порта источника в сообщении может измениться на значение, отличное от 4500. В этом случае ответчик получает сообщение с произвольным портом источника X и портом назначения 4500. После получения этого сообщения все последующие сообщения ответчик отправляет с использованием порта источника 4500 и порта назначения X .

Для отправки UDP-инкапсулированного трафика ESP используются те же номера портов, что и в пакетах IKE. Для того чтобы можно было отличать UDP-инкапсулированный трафик ESP от трафика IKE, в каждое сообщение IKE добавляется маркер *non-ESP*. Это маркер длиной 4 байта со значением 0.

3.3.5. Определение жизнеспособности IPSec-соединения

Зачастую бывает необходимо как можно скорее определить, что соединение с противоположной стороной потеряно. IKE не предоставляет способа выполнить это, кроме как ждать до тех пор, пока не истечет период обновления ключей (lifetime). В большинстве случаев это неприемлемо, поэтому необходим способ проверки состояния противоположной стороны. В этом случае обычно используют механизм *Keepalive* или *Heartbeat*.

В схеме Keepalive каждой стороне требуется регулярное подтверждение жизнеспособности противоположной стороны. Обмен сообщениями происходит с использованием аутентифицированной нагрузки Notify. Оба участника согласовывают на фазе I интервал, в течение которого посылаются сообщения, например 10 секунд. Каждое сообщение HELLO служит доказательством жизнеспособности противоположной стороны. В свою очередь каждая из сторон должна подтвердить сообщение HELLO. Если по истечении 10 секунд какая-либо из сторон не получила HELLO, она сама посыпает сообщение HELLO, ожидая ACK от противоположной стороны в качестве доказательства жизнеспособности. При получении как HELLO, так и ACK таймер перезапускается.

Схема Heartbeat основана на отправке односторонних (без подтверждения) сообщений. Сторона, которая заинтересована в жизнеспособности противоположной стороны, должна полагаться на то, что противоположная сторона сама периодически будет посыпать сообщения HELLO, демонстрируя свою жизнеспособность.

Протокол *Dead Peer Detection (DPD)* описан в RFC 3706 (рис. 3.75). Он позволяет ликвидировать недостатки схем Keepalive и Heartbeat введением определенной логики управления обменом сообщений. В частности, Keepalive и Heartbeat должны обязательно обмениваться HELLO через определенные интервалы времени. При использовании DPD каждая из сторон в меньшей степени зависит от другой. Любая сторона может запрашивать доказательство жизнеспособности в тот момент, когда ей это необходимо, а не через фиксированный интервал времени. Такая асинхронность DPD-обменов позволяет посыпать разное количество сообщений, чем достигается большая масштабируемость.

Рассмотрим взаимодействие двух DPD-узлов *A* и *B*. Если между ними существует IPSec-трафик, то нет необходимости в регулярном доказательстве жизнеспособности. С другой стороны, если существует период простоя, в течение которого между ними нет обмена пакетами, то жизнеспособность каждого узла может вызывать сомнения. Однако знание о жизнеспособности противоположной стороны необходимо только в том случае, если должен быть передан трафик. Например, если узел *A* должен послать IPSec-пакеты после определенного периода простоя, он должен быть уверен в жизнеспособности *B*. В этот момент узел *A* может инициировать DPD-обмен.

DPD-обмен представляет собой двунаправленный обмен сообщениями HELLO/ACK. Сторона, заинтересованная в получении сведений о жизнеспособности противоположной стороны, отправляет запрос R-U-THERE, соответствующий HELLO. Противоположная сторона должна отправить сообщение R-U-THERE-ACK, соответствующее ACK. Получение этого сообщения служит доказательством жизнеспособности.

Следует отметить, что решение о том, когда необходимо инициировать DPD-обмен, во многом зависит от реализации. Возможна реализация, когда DPD-сообщения посыпаются через определенные интервалы после периода простоя.

Каждая сторона может иметь разные требования к определению жизнеспособности. Узлу *A*, например, может требоваться высокая отказоустойчивость, в то время как требования к ресурсам не столь жесткие. При использовании DPD каждая сторона может определить свою собственную «метрику волнения» — интервал, который определяет необходимость DPD-обмена. Этот интервал настраивает администратор на устройстве.

Допустим, что на узле *A* настроили DPD-интервал, равный 10 секундам. Когда узел *A* посыпает исходящий IPSec-трафик, но в течение 10 секунд происходит сбой при получении входящего трафика, она может инициировать DPD-обмен.

С другой стороны, на узле *B* настроили DPD-интервал, равный 5 минутам. Если IPSec-сессия простоявает в течение 5 минут, то узел *B* может инициировать DPD-обмен, если в следующий момент он собирается послать IPSec-пакет к *A*.

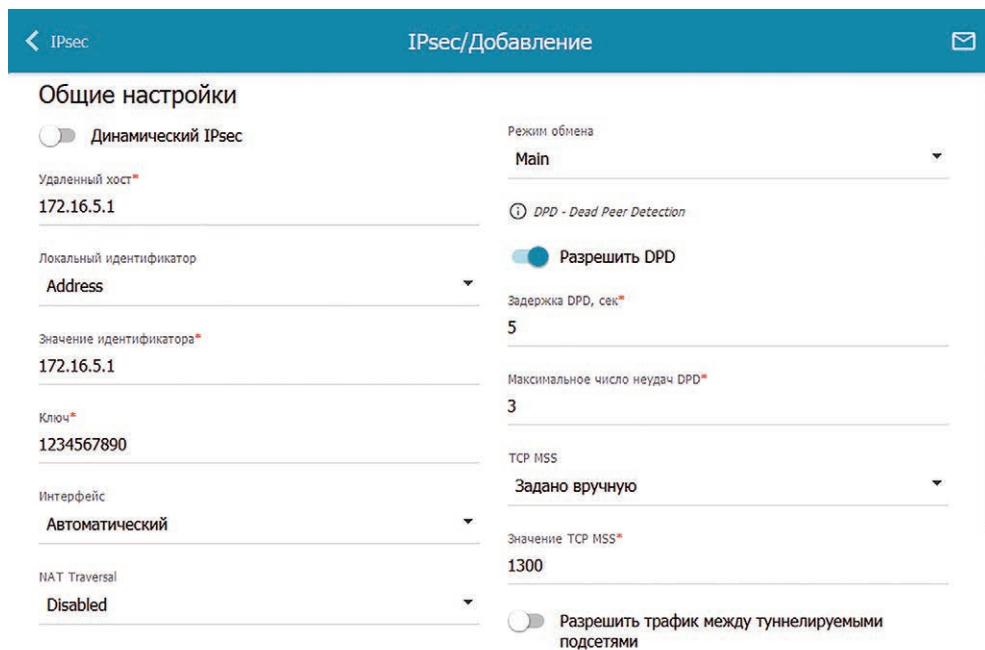


Рис. 3.75. Настройка параметров DPD на маршрутизаторе D-Link

Реализация допускает повторную отправку запроса R-U-THERE, если не получен ответ R-U-THERE-ACK. Администратор может настроить на устройстве количество повторно отправляемых запросов. Если противоположная сторона не отвечает на указанное количество запросов, она считается нежизнеспособной, соединение обрывается, IKE SA и IPsec SA удаляются.

4. Протоколы разрешения адресов

Передача данных через составную сеть выполняется на сетевом уровне модели OSI с использованием IP-адреса, но фактическая передача выполняется на канальном уровне, который использует адреса канального уровня, например MAC-адреса.

В этой главе будет объяснено, каким образом по известному сетевому адресу (IP-адресу) можно узнать адрес канального уровня (MAC-адрес) устройства-получателя.

Два из семи уровней модели OSI ответственны за функции адресации. Это канальный и сетевой уровни. В модели TCP/IP это уровни доступа к сети и к Интернету.

Рассмотрим пример: клиент локальной сети обращается к серверу www.dlink.ru (рис. 4.1).

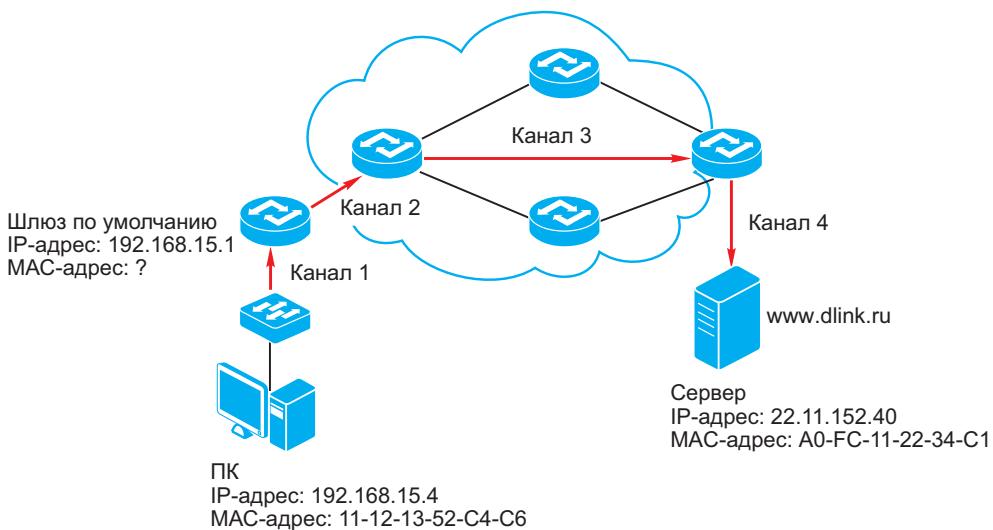


Рис. 4.1. Передача данных в составной сети

Логически соединение осуществляется между клиентом и сервером. Фактически клиент и сервер соединены последовательностью каналов связи, работающих по технологиям канального уровня. В примере таких каналов четыре. Они соединяют между собой маршрутизаторы, находящиеся между клиентом и сервером.

На каждом шаге на основе IP-адреса получателя принимается решение, куда отправить данные, но фактическая передача выполняется на втором уровне, с использованием адреса канального уровня следующего предполагаемого получателя кадра на маршруте.

Так как взаимодействующие устройства соединяются логически, используя IP-адрес, а фактическая передача происходит с использованием адресов канального уровня, возникает вопрос: как узнать адрес канального уровня получателя? Нужен механизм трансляции между адресами этих двух уровней.

Процесс, позволяющий определить адрес канального уровня, используя известный адрес сетевого уровня, называется разрешением адресов (*address resolution*).

Процесс разрешения адресов может выполняться двумя способами:

- С помощью непосредственной привязки (direct mapping);
- С помощью динамического разрешения адресов (dynamic resolution).

При непосредственной привязке MAC-адрес получается из IP-адреса по определенному алгоритму. Этот метод практически не используется в силу ряда ограничений.

Динамическое разрешение адресов выполняется с помощью специального протокола. Устройство, которое знает только сетевой адрес другого устройства, может использовать этот протокол, чтобы узнать его адрес канального уровня.

4.1. Протокол ARP

В стеке TCP/IP для разрешения адресов IPv4 используется протокол ARP. Он определен в RFC 826, опубликованном в 1982 году.

Процесс разрешения адресов в протоколе ARP выполняется путем обмена сообщениями типа «запрос-ответ»:

- ARP Request (ARP-запрос): источник (устройство, которому требуется отправить IP-пакет) посыпает широковещательный запрос всем устройствам локальной сети, чтобы определить, кто является получателем пакета;
- ARP Reply (ARP-ответ): устройство-получатель отправляет назад источнику одноадресное сообщение, сообщая в нем свой адрес канального уровня.

Как и любой другой протокол, ARP имеет специальный формат сообщений, содержащий информацию, требуемую для разрешения адресов. Формат сообщения ARP довольно простой. Он включает поля, описывающие тип сообщения, и адреса канального и сетевого уровней (рис. 4.2).

• *Hardware Type (HTYPE)*: указывает тип транспортного протокола, используемого для передачи сообщения ARP. Например, для Ethernet значение равно 0x0001;

• *Protocol Type (PTYPE)*: указывает тип адреса сетевого уровня, используемого в сообщении ARP. Для IPv4 значение равно 2048 (0800 hex);

• *Hardware Length (HLEN)*: указывает длину физического адреса в байтах. MAC-адреса имеют длину 6 байт;

• *Protocol Length (PLEN)*: указывает длину адреса сетевого уровня в байтах. Адреса IPv4 имеют длину 4 байта;

• *Operation code (Opcode)*: код операции отправителя: 1 — в случае запроса и 2 — в случае ответа;

4. Протоколы разрешения адресов

- *Sender Hardware Address (SHA)*: физический адрес отправителя;
- *Sender Protocol Address (SPA)*: адрес сетевого уровня отправителя;
- *Target Hardware Address (THA)*: физический адрес получателя. Поле пусть при запросе;
- *Target Protocol Address (TPA)*: адрес сетевого уровня с получателя.

: 0	7	15	31
Hardware type (HTYPE)		Protocol type (PTYPE)	
Hardware length (HLEN)	Protocol length (PLEN)	Operation code (Opcode)	
Sender hardware address (SHA)			
Sender protocol address (SPA)			
Target hardware address (THA)			
Target protocol address (TPA)			

Рис. 4.2. Формат протокола ARP

Поскольку ARP является протоколом динамического разрешения адресов, каждое разрешение адресов требует обмена сообщениями по сети. Каждый раз, когда устройство отправляет ARP-сообщение, оно использует полосу пропускания сети, а также загружает ЦПУ сетевых устройств на его обработку.

Решением данной проблемы является использование (*caching*). ARP-кеш представляет собой таблицу, связывающую между собой MAC- и IP-адреса узлов. По сути, таблица — это просто разделы оперативной памяти устройства.

Каждое устройство в сети создает и обслуживает свою собственную таблицу ARP. Существуют два способа создания записей в таблице ARP:

- Статические записи (Static ARP Cache Entries): записи, связывающие физические адреса с IP-адресами создаются вручную и постоянно хранятся в таблице ARP. Статические записи можно использовать в том случае, если устройства взаимодействуют на постоянной основе;
- Динамические записи (Dynamic ARP Cache Entries): связки физический адрес/IP-адрес, создаются динамически в результате работы протокола ARP. Они хранятся в таблице только определенный период времени и затем удаляются. Это делается для того, чтобы не использовать много системной памяти и записи в таблице были актуальными;

В таблице ARP могут храниться как статические, так и динамические записи. На рис. 4.3 и рис. 4.4 показаны ARP-таблицы коммутатора и подключенного к нему компьютера.

Администратор сети может управлять ARP-таблицей устройств, в которых реализован стек TCP/IP: просматривать записи, создавать статические записи, удалять записи, устанавливать время жизни динамических записей.

```
C:\Users\edu1>arp -a
```

Интерфейс: 192.168.15.2 --- 0xb	адрес в Интернете	Физический адрес	Тип
	192.168.15.1	ac-f1-df-b5-fc-00	динамический
	192.168.15.255	ff-ff-ff-ff-ff-ff	статический
	224.0.0.2	01-00-5e-00-00-02	статический
	224.0.0.22	01-00-5e-00-00-16	статический
	224.0.0.252	01-00-5e-00-00-fc	статический
	239.255.255.250	01-00-5e-7f-ff-fa	статический

Рис. 4.3. ARP-таблица компьютера

```
DGS-3120-24TC:admin#show arpentry
Command: show arpentry
```

ARP Aging Time : 20

Interface	IP Address	MAC Address	Type
System	192.168.15.0	FF-FF-FF-FF-FF-FF	Local/Broadcast
System	192.168.15.1	AC-F1-DF-B5-FC-00	Local
System	192.168.15.2	00-33-22-33-44-55	Dynamic
System	192.168.15.255	FF-FF-FF-FF-FF-FF	Local/Broadcast

Total Entries: 4

Рис. 4.4. ARP-таблица коммутатора

На рабочих станциях это выполняется с помощью утилиты arp. Управление ARP-таблицей сетевых устройств выполняется с использованием специальных команд, определенных в их программном обеспечении.

4.1.1. Операции ARP

Давайте рассмотрим, какие действия выполняются протоколом ARP в процессе разрешения адресов. Когда отправитель определил IP-адрес получателя, он обращается к своей ARP-таблице, чтобы узнать его MAC-адрес. Если отправитель находит в таблице запись, связывающую MAC- и IP-адрес получателя, то он получает информацию о MAC-адресе, формирует кадр и отправляет его в сеть.

4. Протоколы разрешения адресов

Если в таблице отсутствует запись, связывающая MAC- и IP-адрес получателя, отправитель формирует ARP-запрос (рис. 4.5). Поле Target MAC address запроса заполняется нулями, в поле Target IP address указывается IP-адрес устройства, чей MAC-адрес надо узнать. На канальном уровне этот запрос помещается в кадр и отправляется в сеть. В качестве адреса источника в кадре указывается уникальный MAC-адрес отправителя. В качестве адреса приемника — широковещательный MAC-адрес FF:FF:FF:FF:FF:FF.

No.	Time	Source	Destination	Protocol	Length	Info
325	301.983726	Universa_0a:9c:e1	Broadcast	ARP	42	who has 192.168.100.1? tell 192.168.100.2
Frame 325: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)						
Ethernet II, Src: Universa_0a:9c:e1 (cc:52:af:0a:9c:e1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)						
Address Resolution Protocol (request)						
Hardware type: Ethernet (1) Protocol type: IP (0x0800) Hardware size: 6 Protocol size: 4 Opcode: request (1) [Is gratuitous: False] Sender MAC address: Universa_0a:9c:e1 (cc:52:af:0a:9c:e1) Sender IP address: 192.168.100.2 (192.168.100.2) Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00) Target IP address: 192.168.100.1 (192.168.100.1)						

Рис. 4.5. ARP-запрос

Поскольку кадр широковещательный, его принимают все устройства локальной сети. Далее они извлекают из него и анализируют ARP-запрос. Если IP-адрес устройства-получателя и IP-адрес, содержащийся в поле Target IP address ARP-запроса, не совпадают, запрос отбрасывается. В противном случае формируется ARP-ответ (рис. 4.6), в поле Sender MAC address которого указывается запрашиваемый MAC-адрес. ARP-ответ помещается в одноАдресный кадр. В качестве адреса источника в нем указан MAC-адрес устройства-получателя, а в качестве адреса назначения — MAC-адрес отправителя ARP-запроса.

После получения ARP-запроса устройство-получатель обновляет свою ARP-таблицу. В нее заносится запись о связке MAC- и IP-адресов его отправителя. Это позволяет оптимизировать процесс разрешения адресов и избавиться от отправки лишних запросов.

No.	Time	Source	Destination	Protocol	Length	Info
325	301.983726	Universa_0a:9c:e1	Broadcast	ARP	42	who has 192.168.100.1? tell 192.168.100.2
326	301.985251	94:04:9c:64:b8:ba	Universa_0a:9c:e1	ARP	42	192.168.100.1 is at 94:04:9c:64:b8:ba
Frame 326: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)						
Ethernet II, Src: 94:04:9c:64:b8:ba (94:04:9c:64:b8:ba), Dst: Universa_0a:9c:e1 (cc:52:af:0a:9c:e1)						
Address Resolution Protocol (reply)						
Hardware type: Ethernet (1) Protocol type: IP (0x0800) Hardware size: 6 Protocol size: 4 Opcode: reply (2) [Is gratuitous: False] Sender MAC address: 94:04:9c:64:b8:ba (94:04:9c:64:b8:ba) Sender IP address: 192.168.100.1 (192.168.100.1) Target MAC address: Universa_0a:9c:e1 (cc:52:af:0a:9c:e1) Target IP address: 192.168.100.2 (192.168.100.2)						

Рис. 4.6. ARP-ответ

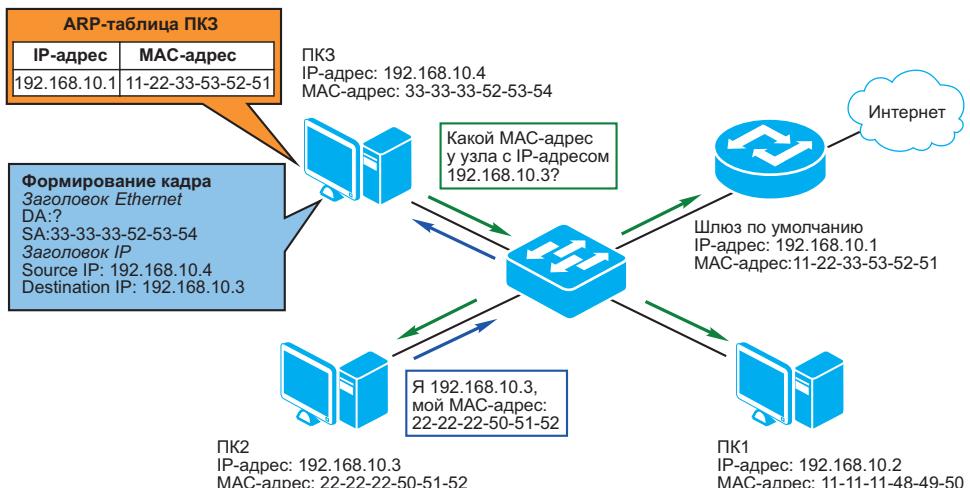


Рис. 4.7. Процесс разрешения адресов

Когда отправитель ARP-запроса получает ответ, он обновляет свою ARP-таблицу. Таким образом, ему становится известен MAC-адрес получателя и процесс разрешения адресов (рис. 4.7) завершается.

4.2. Gratuitous ARP

Узел может использовать протокол ARP для оповещения других устройств сети о своей привязке адресов MAC-IP. Такие оповещения получили название **Gratuitous ARP** (беспричинный ARP), т. е. ответ на предполагаемый вопрос, которого на самом деле не было.

Gratuitous ARP представляет собой или пакет ARP Request, или пакет ARP Reply, который широковещательно отправляется узлом, чтобы уведомить другие узлы о необходимости обновить записи в их ARP-таблицах. При использовании Gratuitous ARP Request (рис. 4.8) в полях Sender IP Address и Target IP Address указывается IP-адрес отправителя, в поле Sender MAC Address указывается MAC-адрес отправителя, в поле Target MAC Address — широковещательный адрес FF:FF:FF:FF:FF:FF. Ответ на этот запрос не отправляется, так как узел отправляет запрос для разрешения своего собственного IP-адреса.

Когда используется пакет Gratuitous ARP Reply в полях Sender IP Address и в Target IP Address указывается IP-адрес отправителя, в полях Sender MAC Address и Target MAC Address указывается MAC-адрес отправителя. Пакет Gratuitous ARP Reply не отправляется в ответ на какой-либо запрос.

В сети Gratuitous протокол ARP может использоваться:

- Для определения дублирования IP-адресов. В правильно сконфигурированной сети на пакет Gratuitous ARP Request не будет ответа. Однако если другой узел в сети сконфигурирован с таким же IP-адресом, как у отправителя, он получит пакет ARP Reply от этого узла;

4. Протоколы разрешения адресов

No.	Time	Source	Destination	Protocol	-	Length	Info
1	0.000000000	84:c9:b2:1f:9c:00	Broadcast	ARP	-	60	Gratuitous ARP for 10.90.90.91 (Request)
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) Ethernet II, Src: 84:c9:b2:1f:9c:00 (84:c9:b2:1f:9c:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff) Address Resolution Protocol (request/gratuitous ARP) Hardware type: Ethernet (1) Protocol type: IP (0x0800) Hardware size: 6 Protocol size: 4 Opcode: request (1) [Is gratuitous: True] Sender MAC address: 84:c9:b2:1f:9c:00 (84:c9:b2:1f:9c:00) Sender IP address: 10.90.90.91 (10.90.90.91) Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff) Target IP address: 10.90.90.91 (10.90.90.91)							

Рис. 4.8. Пакет Gratuitous ARP Request

- Оповещать, что IP-адрес привязан к новому MAC-адресу. Если на узле заменили сетевой адаптер, то привязка адресов IP-MAC также изменится. После перезагрузки системы она отправит пакет ARP Request на свой собственный IP-адрес. Все остальные узлы сети получат и обработают этот пакет. Каждый узел обновит запись, относящуюся к этому IP-адресу, в своей ARP-таблице, если она уже существовала в ней. При этом если в ARP-таблице устройства отсутствовала запись, связанная с IP-адресом из пакета Gratuitous ARP, новая запись создана не будет;

- Уведомлять коммутаторы, что узел был переключен с одного порта на другой.

ARP spoofing

Протокол ARP не реализует методов защиты и проверки подлинности пакетов. ARP spoofing является одним из методов инициирования атаки типа «man-in-the-middle». Атакующий перехватывает в сети ARP-запрос и отправляет ложный ARP-ответ, в котором объявляет себя искомым устройством локальной сети (указывает в нем MAC-адрес ложного устройства и IP-адрес реального). Коммутатор получит ложный ARP-ответ и обновит ARP-таблицу. Таким образом, вместо того чтобы отправлять трафикциальному устройству, коммутатор начнет перенаправлять его на устройство злоумышленника. В итоге правильный пункт назначения не получит данные, а злоумышленник сможет перехватывать пароли, ключи и другую информацию.

Одним из типов ARP spoofing является использование пакетов Gratuitous ARP, так как эти пакеты не требуют запросов или ответов. Злоумышленник «отправляет» ARP-таблицу коммутатора, отправляя Gratuitous ARP Request, в котором для определенного IP-адреса указан ложный MAC-адрес. После получения злонамеренного пакета Gratuitous ARP Request запись для этого IP-адреса в ARP-таблице коммутатора обновляется.

Для защиты от подобных атак на коммутаторах поддерживается механизм Dynamic ARP Inspection (DAI). Когда функция DAI настроена на коммутаторе, он перехватывает входящие ARP-пакеты и проверяет их параметры IP-MAC на соответствие записям, связывающим все разрешенные

IP- и MAC-адреса сети, которые созданы вручную администратором или динамически с помощью механизма DHCP spoofing. Пакеты ARP, в которых не будет совпадать хотя бы один параметр, будут отбрасываться.

4.3. Proxy ARP

Протокол ARP был разработан для использования устройствами в пределах одной локальной сети. Однако возможна ситуация, когда узлы находятся в разных физических сетях. В этом случае узлы из разных физических сетей не могут получать широковещательные ARP-запросы и отвечать на них. Если сеть разбита на подсети и они соединены маршрутизатором, то он будет видеть ARP-запросы, отправляемые узлами из одной подсети узлам в другой. Однако маршрутизаторы не пропускают широковещательный трафик и устройства, находящиеся в разных подсетях не смогут выполнить разрешение адресов.

Для решения этой проблемы используется метод *Proxy ARP*, описанный в RFC 1027. Этот механизм применяется в маршрутизаторах или коммутаторах L3, чтобы отвечать на ARP-запросы, предназначенные для другого устройства. Когда маршрутизатор (коммутатор L3) с активизированной функцией Proxy ARP получает ARP-запрос для разрешения IP-адреса, находящегося в другой подсети, он посыпает ARP-ответ, содержащий его собственный MAC-адрес. Узел, получивший этот ответ, будет пересыпать все пакеты, которые должны достигнуть требуемого адресата на маршрутизатор (коммутатор L3). Так как маршрутизатору (коммутатору L3) известен путь к требуемому узлу, он будет перенаправлять ему все пакеты.

Proxy ARP обычно используется в тех сетях, где на IPv4-узлах не настроен шлюз по умолчанию или они не обладают функциями маршрутизации. В настоящее время этот механизм практически не используется, поскольку содержит значительные недостатки. Он увеличивает объем широковещательного ARP-трафика, размеры ARP-таблиц устройств, может использоваться для атак ARP spoofing. Поэтому для передачи пакетов между различными подсетями, соединенными маршрутизирующим устройством наилучшим решением является использование протоколов маршрутизации.

4.4. Разрешение адресов для IPv6

Изменения, которые были сделаны в IPv6, коснулись не только самого протокола IP, но и всех протоколов стека TCP/IP. В IPv6 функция разрешения адресов, которую выполняет ARP, объединена с несколькими функциями, которые выполняются протоколом ICMP в оригинальном стеке TCP/IP, расширена дополнительными возможностями и реализована в протоколе *Neighbor Discovery Protocol* (NDP). Протокол NDP описан в RFC 4861.

Базовая концепция разрешения адресов в IPv6 осталась прежней: разрешение адресов является динамическим и основано на использовании таблицы, связывающей между собой MAC- и IP-адреса.

4. Протоколы разрешения адресов

Когда устройство хочет отправить пакет IPv6 соседнему устройству в локальной сети, но не знает его физический адрес, оно инициирует процесс разрешения адресов, но посыпает не широковещательный ARP-запрос, а сообщение Neighbor Solicitation (рис. 4.9) протокола NDP.

Если нижележащий протокол канального уровня поддерживает многоадресную (групповую) рассылку, как, например, Ethernet, сообщение Neighbor Solicitation не отправляется широковещательно. Оно отправляется на групповой адрес Solicited-Node того устройства, чей адрес IPv6 необходимо разрешить.

No.	Time	Source	Destination	Protocol	Length	Info
67	7.837652000	fe80::f4af:b8c7:5c12:46	ff02::1:ff1f:9c00	ICMPv6	86	Neighbor Solicitation for fe80::86c9:b2ff:fe1f:9c00
Frame 67: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)						
Ethernet II, Src: 00:33:22:33:44:55 (00:33:22:33:44:55), Dst: IPv6icast_ff:1f:9c:00 (33:33:ff:1f:9c:00)						
Internet Protocol Version 6, Src: fe80::f4af:b8c7:5c12:464 (fe80::f4af:b8c7:5c12:464), Dst: ff02::1:ff1f:9c00 (ff02::1:ff1f:9c00)						
@ 0110 = Version: 6						
@ 0000 0000 = Traffic class: 0x00000000						
.... 0000 0000 0000 0000 = Flowlabel: 0x00000000						
Payload length: 32						
Next header: ICMPv6 (0x3a)						
Hop limit: 255						
Source: fe80::f4af:b8c7:5c12:464 (fe80::f4af:b8c7:5c12:464)						
Destination: ff02::1:ff1f:9c00 (ff02::1:ff1f:9c00)						
@ Internet Control Message Protocol v6						
Type: Neighbor Solicitation (135)						
Code: 0						
Checksum: 0x97ea [correct]						
Reserved: 00000000						
Target Address: fe80::86c9:b2ff:fe1f:9c00 (fe80::86c9:b2ff:fe1f:9c00)						
@ ICMPv6 Option (Source link-layer address : 00:33:22:33:44:55)						
Type: Source link-layer address (1)						
Length: 1 (8 bytes)						
Link-layer address: 00:33:22:33:44:55 (00:33:22:33:44:55)						

Рис. 4.9. Сообщение Neighbor Solicitation

При получении сообщения Neighbor Solicitation устройство назначения отправит назад устройству-отправителю сообщение Neighbor Advertisement (аналогично ARP Reply). Поскольку адрес Solicited-Node не является уникальным, устройство-получатель должно убедиться, что оно является тем устройством, чей адрес пытается разрешить устройство-отправитель. Подробнее протокол NDP будет рассмотрен далее в нашем курсе.

No.	Time	Source	Destination	Protocol	Length	Info
68	7.839846000	fe80::86c9:b2ff:fe1f:9c fe80::f4af:b8c7:5c12:46	ICMPv6	86	Neighbor Advertisement fe80::86c9:b2ff:fe1f:9c00	
Frame 68: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)						
Ethernet II, Src: 84:c9:b2:1f:9c:00 (84:c9:b2:1f:9c:00), Dst: 00:33:22:33:44:55 (00:33:22:33:44:55)						
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c00 (fe80::86c9:b2ff:fe1f:9c00), Dst: fe80::f4af:b8c7:5c12:464 (fe80::f4af:b8c7:5c12:464)						
@ 0110 = Version: 6						
@ 0000 0000 = Traffic class: 0x00000000						
.... 0000 0000 0000 0000 = Flowlabel: 0x00000000						
Payload length: 32						
Next header: ICMPv6 (0x3a)						
Hop limit: 255						
Source: fe80::86c9:b2ff:fe1f:9c00 (fe80::86c9:b2ff:fe1f:9c00)						
[Source SA MAC: 84:c9:b2:1f:9c:00 (84:c9:b2:1f:9c:00)]						
Destination: fe80::f4af:b8c7:5c12:464 (fe80::f4af:b8c7:5c12:464)						
@ Internet Control Message Protocol v6						
Type: Neighbor Advertisement (136)						
Code: 0						
Checksum: 0x1175 [correct]						
Flags: 0xe0000000						
Target Address: fe80::86c9:b2ff:fe1f:9c00 (fe80::86c9:b2ff:fe1f:9c00)						
@ ICMPv6 Option (Target link-layer address : 84:c9:b2:1f:9c:00)						
Type: Target link-layer address (2)						
Length: 1 (8 bytes)						
Link-layer address: 84:c9:b2:1f:9c:00 (84:c9:b2:1f:9c:00)						

Рис. 4.10. Сообщение Neighbor Advertisement

5. Протокол ICMP

Основным протоколом сетевого уровня является протокол IP, который позволяет доставлять данные в сетях TCP/IP между узлами составной сети. Протокол IP не гарантирует надежной доставки пакета до адресата. Другими словами, отправитель передает пакеты через составную сеть и не получает подтверждение доставки.

Протокол IP не обладает механизмами отправки служебных сообщений, которые позволяли бы сообщать устройству-отправителю о проблемах, возникающих при передаче пакетов, или осуществлять тестирование соединения. Поэтому недостающие в протоколе IP функции выполняются с помощью протокола *Internet Control Message Protocol (ICMP)*.

Протокол ICMP является неотъемлемой частью протокола IP и обеспечивает его поддержку в форме ICMP-сообщений, которые инкапсулируются в IP-пакеты. Первоначальный протокол ICMP был определен в RFC 792 одновременно с протоколом IP, описанным в RFC 791. В середине 1990-х годов появился протокол IPv6, для которого также надо было определить версию протокола ICMP. Таким образом, новую версию протокола ICMP назвали ICMPv6 (RFC 4443), а оригинальную версию ICMP стали называть ICMPv4.

Сообщения ICMPv4 и ICMPv6 имеют одинаковый базовый формат (рис. 5.1). Структура сообщения может быть разделена на две части — общую и уникальную. Общая часть состоит из трех полей, размер которых одинаков для всех типов сообщений:

- *Type (Тип)* — определяет тип сообщения ICMP;
- *Code (Код)* — определяет подтип сообщения внутри ICMP-сообщения каждого типа;
- *Checksum (Контрольная сумма)* — вычисляется аналогично контрольной сумме заголовка IPv4.

Уникальная часть содержит поля, определенные для каждого типа сообщения.

Биты: 0	7	15	31
Type	Code	Checksum	
Содержимое сообщения (зависит от значений полей Type и Code)			

Рис. 5.1. Формат сообщения ICMPv4/v6

Обе версии протокола ICMPv4 и ICMPv6 определяют общую систему передачи сообщений. В дополнение к сообщениям, определенным протоколом ICMP, другие протоколы также могут определять свои типы сообщений ICMP. Некоторые самые важные из них приведены в табл. 5.1.

5. Протокол ICMP

Таблица 5.1. Протоколы, определяющие сообщения ICMP

Версия ICMP	RFC	Название	Определяемые типы сообщений ICMP
ICMPv4	950	Internet Standard Subnetting Procedure	<i>Address Mask Request, Address Mask Reply</i>
	1256	ICMP Router Discovery Messages	<i>Router Advertisement, Router Solicitation</i>
	1393	Traceroute Using an IP Option	<i>Traceroute</i>
	1812	Requirements for IP Version 4 Routers	Определяет три кода (подтипа) для сообщения <i>Destination Unreachable</i>
ICMPv6	2461	Neighbor Discovery for IP Version 6 (IPv6)	<i>Router Advertisement, Router Solicitation, Neighbor Advertisement, Neighbor Solicitation, Redirect</i>
	2894	Router Renumbering for IPv6	<i>Router Renumbering</i>

Протокол ICMP является простейшим протоколом стека TCP/IP и отличается от других его протоколов тем, что не выполняет определенных задач и алгоритмов. Он описывает механизм, с помощью которого могут передаваться и приниматься различные управляющие сообщения, обеспечивая выполнение разнообразных функций. Протокол ICMP определяет различные типы сообщений, которые позволяют устройствам обмениваться информацией. Однако он не определяет, как они используются. Это делают протоколы, использующие сообщения ICMP в своей работе.

ICMP рассматривается как часть IP и использует IP для отправки сообщений. Сообщения отправляются на сетевой уровень принимающего устройства, как показано на рис. 5.2. Предположим, что ПК1 отправляет IP-пакет серверу. Маршрутизатор R2 получает пакет и обнаруживает, что возникла проблема и пакет необходимо отбросить. Для этого маршрутизатор R2 отправляет назад ПК1 ICMP-сообщение с описанием проблемы. Обратите внимание, что R2 отправляет сообщение непосредственно ПК1, а не соседнему маршрутизатору R1.

Протокол ICMP не использует номера портов, как протоколы TCP или UDP, для определения приложения сетевого узла, которому предназначено сообщение. Программное обеспечение стека протоколов TCP/IP, функционирующее на сетевом узле, самостоятельно распознает тип ICMP-сообщения и направляет его соответствующему приложению.

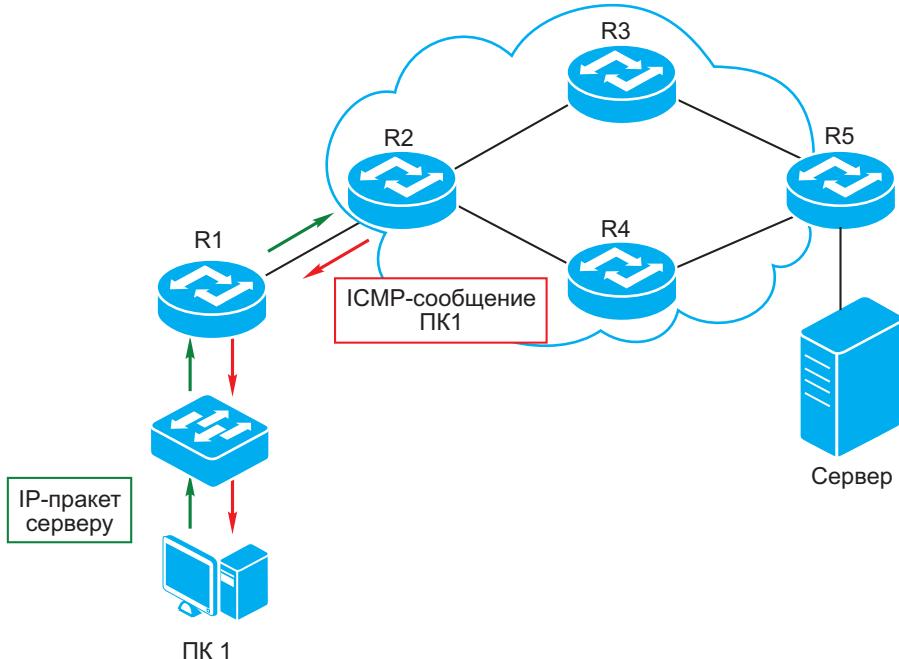


Рис. 5.2. Отправка ICMP-сообщения при возникновении проблем передачи пакета

5.1. Классы, типы и коды сообщений ICMP

Протокол ICMP определяет различные типы сообщений, которые позволяют устройствам обмениваться различной информацией.

Сообщения ICMP делятся на два класса:

- *сообщения об ошибках* — используются для оповещения устройства-отправителя о проблемах, возникших при передаче пакета. Обычно генерируются в ответ на какое-либо событие, возникшее при передаче пакета;
- *информационные сообщения* — используются для диагностики, тестирования, обмена информацией и других целей. Не содержат описания ошибок и обычно не отправляются при возникновении какого-либо события. Могут генерироваться соответствующим приложением либо как ответ на другое информационное сообщение ICMP, либо на постоянной основе для предоставления информации другим устройствам.

Каждому сообщению ICMP присваивается уникальное значение типа, которое указывается в 8-битном поле Type пакета ICMP. Для ICMPv4 и ICMPv6 может быть определено до 256 различных типов сообщений (отдельный набор для каждого протокола). В ICMPv4 значения типа назначаются последовательно для информационных сообщений и сообщений об ошибках. В ICMPv6 сообщениям об ошибках в поле Type присваиваются значения от 0 до 127, информационным сообщениям — от 128 до 255.

5. Протокол ICMP

Тип сообщения указывает на общее предназначение каждого ICMP-сообщения. Коды, указываемые в 8-битном поле Code пакета ICMP, вносят дополнительный уровень детализации. Каждый тип сообщения может иметь до 256 подтипов.

В табл. 5.2 представлены основные типы ICMP-сообщений протоколов ICMPv4 и ICMPv6, показаны значения поля Type, названия сообщений и их краткое описание.

Как видно из таблицы, несколько типов сообщений ICMPv4 и ICMPv6 в целом похожи, но имеют небольшие различия. Например, сообщение Redirect в ICMPv4 рассматривается как сообщение об ошибке, а в ICMPv6 — как информационное сообщение. Способы использования сообщений также различны. Многие сообщения ICMPv6 используются протоколом NDP.

Таблица 5.2. Классы и типы сообщений ICMP

Класс сообщения	Значение Type	Название сообщения	Описание типа сообщения
Сообщения об ошибках ICMPv4	3	<i>Destination Unreachable</i>	Указывает, что пакет не может быть доставлен адресату. Код сообщения указывает причину ошибки
	5	<i>Redirect</i>	Позволяет маршрутизатору информировать узел о наилучшем маршруте передачи пакета
	11	<i>Time Exceeded</i>	Отправляется, когда пакет отбрасывается по причине достижения его полем Time To Live (TTL) значения 0
	12	<i>Parameter Problem</i>	Указывает, что при доставке существуют проблемы с параметрами пакета
Информационные сообщения ICMPv4	0	<i>Echo Reply</i>	Отправляется в ответ на сообщение Echo (Request). Используется для тестирования соединения
	8	<i>Echo Request</i>	Отправляется устройством при тестировании соединения с другим устройством в сети
	9	<i>Router Advertisement</i>	Используется маршрутизаторами для объявления о своем существовании и возможностях
	10	<i>Router Solicitation</i>	Запрос маршрутизатора на отправку Router Advertisement
	30	<i>Traceroute</i>	Используется для трассировки маршрута утилитой traceroute

Окончание табл. 5.2

Класс сообщения	Значение Type	Название сообщения	Описание типа сообщения
Сообщения об ошибках ICMPv6	1	Destination Unreachable	Указывает, что пакет не может быть доставлен адресату. Код сообщения указывает причину ошибки
	2	Packet Too Big	Отправляется, когда пакет не может быть передан, так как слишком большой для MTU следующего маршрутизатора на пути следования. Это сообщение требуется только в IPv6, поскольку в IPv4 маршрутизатор может фрагментировать большие пакеты
	3	Time Exceeded	Отправляется, когда пакет отбрасывается по причине достижения его полем Hop Limit значения 0
	4	Parameter Problem	Указывает, что при доставке существуют проблемы с параметрами пакета
Информационные сообщения ICMPv6	128	Echo Request	Отправляется устройством при тестировании соединения с другим устройством в сети
	129	Echo Reply	Отправляется в ответ на сообщение Echo (Request)
	133	Router Solicitation	Запрос маршрутизатора на отправку Router Advertisement
	134	Router Advertisement	Используется маршрутизаторами для объявления о своем существовании и возможностях
	135	Neighbor Solicitation	Отправляется устройством при запросе адреса канального уровня другого устройства
	136	Neighbor Advertisement	Объявление информации об узле другим устройствам сети
	137	Redirect	Перенаправление передачи от узла ближайшему соседу в сети или маршрутизатору

5.2. Правила генерации сообщений ICMP

Информационные ICMP-сообщения генерируются в соответствии с правилами, установленными использующими их протоколами. ICMP-сообщения об ошибках генерируются в ответ на какое-либо событие, возникшее при передаче пакета. Может возникнуть ситуация, при которой узлы начнут отправлять сообщения об ошибках друг другу, что приведет к петлям или отправке множественных сообщений.

Для предотвращения таких проблем, ICMP-сообщения об ошибках **не должны** генерироваться в ответ на:

- ICMP-сообщения об ошибках;
- IP-пакеты с широковещательным или групповым адресом, чтобы не вызывать перегрузку в сети («широковещательный шторм»);
- фрагменты IP-пакетов за исключением первого — при повреждении фрагментированного IP-пакета ICMP-сообщение отправляется только после получения первого поврежденного фрагмента, поскольку отправитель все равно повторит передачу всего IP-пакета целиком.

Перечисленные правила относятся к сообщениям ICMPv4 и ICMPv6, однако в ICMPv6 определены некоторые исключения. В частности, ICMPv6-сообщение Packet Too Big может генерироваться в ответ на групповой адрес, так как это требуется для работы механизма Path MTU Discovery. Некоторые сообщения Parameter Problem также могут отправляться на широковещательные или групповые адреса.

5.3. Утилита Ping

Ping (Packet InterNet Groper) — это утилита для проверки соединений (рис. 5.3) в сетях на основе TCP/IP, которая отправляет запрос ICMP Echo Request указанному узлу и ожидает от него ответ ICMP Echo Reply. Утилита **ping** является одним из основных диагностических средств в сетях TCP/IP.

Для проверки соединения с требуемым узлом используется команда:

ping <IP addr> или **ping <доменное имя>**.

Когда утилита запускается без использования дополнительных опций, то для таких параметров как размер отправляемого сообщения, число отправляемых запросов, время ожидания ответа и др., используются значения по умолчанию.

```
c:\>ping 192.168.100.1

Обмен пакетами с 192.168.100.1 по с 32 байтами данных:
Ответ от 192.168.100.1: число байт=32 время=1мс TTL=64
Ответ от 192.168.100.1: число байт=32 время=2мс TTL=64
Ответ от 192.168.100.1: число байт=32 время=1мс TTL=64
Ответ от 192.168.100.1: число байт=32 время=1мс TTL=64

Статистика Ping для 192.168.100.1:
Пакетов: отправлено = 4, получено = 4, потеряно = 0
(0% потеря)
Приблизительное время приема-передачи в мс:
    Минимальное = 1мсек, Максимальное = 2 мсек, Среднее = 1 мсек
```

Рис. 5.3. Проверка соединения с помощью утилиты ping в сети IPv4

```
C:\Windows\System32>ping fdd0:1:6:0:9055:32c6:d34b:a802

Обмен пакетами с fdd0:1:6:0:9055:32c6:d34b:a802 по с 32 байтами данных:
Ответ от fdd0:1:6:0:9055:32c6:d34b:a802: время=1мс
Ответ от fdd0:1:6:0:9055:32c6:d34b:a802: время=1мс
Ответ от fdd0:1:6:0:9055:32c6:d34b:a802: время=1мс
Ответ от fdd0:1:6:0:9055:32c6:d34b:a802: время=1мс

Статистика Ping для fdd0:1:6:0:9055:32c6:d34b:a802:
Пакетов: отправлено = 4, получено = 4, потеряно = 0
<0% потеря>
Приблизительное время приема-передачи в мс:
Минимальное = 1мсек, Максимальное = 1 мсек, Среднее = 1 мсек
```

Рис. 5.4. Проверка соединения с помощью утилиты ping в сети IPv6

Утилита будет отправлять требуемому узлу серию запросов Echo Request и сообщать, на каждый ли запрос получен ответ Echo Reply. Наличие ответа Echo Reply означает, что удаленный узел включен, находится в рабочем состоянии и все функции физического, канального и сетевого уровней на нем работают нормально. При этом проверка любой функциональности вышеуказанных уровней не выполняется. Отсутствие ответа на запрос Echo Request может быть связано с блокировкой ICMP-сообщений межсетевым экраном.

Если ответ получен, утилита сообщает, сколько времени прошло между отправкой запроса и получением ответа. После завершения работы утилиты покажет итоговую статистику об отправленных, принятых и потерянных пакетах, а также их среднее время приема-передачи (рис. 5.4). С помощью этой статистики можно оценить качество канала (загруженность на канале и промежуточных устройствах).

В своей простейшей форме, т. е. без дополнительных опций, утилита ping может использоваться для следующих видов диагностики:

- *Проверка работы стека TCP/IP устройства*: выполняя команду ping на собственный IP-адрес устройства или адрес интерфейса обратной петли (loopback), можно проверить работу стека TCP/IP на устройстве;

- *Проверка подключения по локальной сети*: выполняя команду ping на IP-адрес устройства локальной сети, можно проверить соединение с ним;

- *Проверка работы шлюза по умолчанию*: выполняя команду ping на IP-адрес интерфейса маршрутизатора, на который перенаправляется весь трафик, не предназначенный для устройств данной локальной сети, можно проверить его работу и доступность;

- *Проверка функциональности системы DNS*: если результатом выполнения команды ping на доменное имя явилось сообщение о том, что данный узел не удалось обнаружить, попытайтесь выполнить ping на IP-адрес этого узла. Если ответы от этого узла были получены, значит, существует проблема с конфигурацией или разрешением доменных имен;

- *Проверка работы удаленного узла*: если в результате выполнения команды ping на IP-адрес удаленного узла ответы не были получены, можно попытаться проверить соединение с другим удаленным узлом. Если ответы

5. Протокол ICMP

будут получены, значит, существует проблема с удаленным, а не локальным устройством.

Утилита ping включает ряд опций и параметров, которые может настраивать администратор. Это позволяет использовать утилиту для более тщательного тестирования. Например, в ОС Windows команда ping может быть выполнена с параметром -t, который позволит выполнять обмен Echo-пакетами непрерывно (до ее административного прекращения), что удобно использовать для выявления периодически возникающих в течение долгого периода времени проблем (рис. 5.5).

```
C:\>ping

Использование:
ping [-t] [-a] [-n <число>] [-l <размер>] [-f] [-i <TTL>] [-v <TOS>]
      [-r <число>] [-s <число>] [[-j <список узлов>] | [-k <список узлов>]]
      [-w <тайм-аут>] [-R] [-S <адрес источника>] [-4] [-6] конечный_узел

Параметры
-t          Проверка связи с указанным узлом до прекращения.
            Для отображения статистики и продолжения проверки
            нажмите сочетание клавиш CTRL+BREAK;
            для прекращения нажмите CTRL+C.
-a          Определение имен узлов по адресам.
-n <число> Число отправляемых запросов эха.
-l <размер> Размер буфера отправки.
-f          Установка в пакете флага, запрещающего
            фрагментацию (только IPv4).
-i <TTL>   Задание срока жизни пакетов.
-v <TOS>   Задание типа службы (только IPv4. Этот параметр
            недоступен и не влияет на поле TOS в заголовке IP).
-r <число> Запись маршрута для указанного числа прыжков
            (только IPv4).
-s <число> Отметка времени для указанного числа прыжков
            (только IPv4).
-j <список_узлов> Свободный выбор маршрута по списку узлов
            (только IPv4).
-k <список_узлов> Жесткий выбор маршрута по списку узлов
            (только IPv4).
-w <тайм-аут> Тайм-аут для каждого ответа (в миллисекундах).
-R          Использование заголовка для проверки также и
            обратного маршрута (только IPv6).
-S <адрес источника> Используемый адрес источника.
-4          Принудительное использование протокола IPv4.
-6          Принудительное использование протокола IPv6.
```

Рис. 5.5. Опции и параметры утилиты ping в ОС Windows

6. Протокол NDP

Изменения, которые были сделаны в IPv6, коснулись не только самого протокола IP, но и служебных протоколов сетевого уровня. В частности, в стеке TCP/IPv4 для разрешения адресов канального уровня используется протокол ARP. В стеке TCP/IPv6 функция разрешения адресов и ряд функций, относящихся к взаимодействию устройств в локальной сети, реализованы протоколом *NDP* (*Neighbor Discovery Protocol — протокол обнаружения соседей*). Протокол NDP, подобно протоколу ICMP, является протоколом обмена сообщениями, с помощью которых он выполняет ряд функций. В настоящее время он определен в RFC 4861.

Понятие «сосед» используется в различных сетевых протоколах и технологиях для обозначения устройств, способных отправлять сообщения непосредственно друг другу. В локальной сети имеются как компьютеры, так и маршрутизаторы (коммутаторы L3), поэтому термин «сосед» может применяться к любому из устройств. Поскольку компьютеры и маршрутизаторы играют разные роли в сети, в результате для этих устройств процесс обнаружения соседей будет различаться.

В RFC 4861 определены девять функций, выполняемых протоколом NDP. Для ясности эти функции можно разбить на три группы, как показано на рис. 6.1.

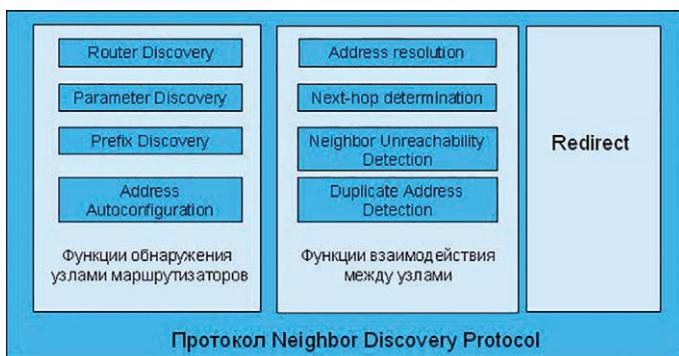


Рис. 6.1. Функции, выполняемые протоколом NDP

Задачу обнаружения в локальной сети маршрутизаторов и обмена данными между ними и узлами выполняют четыре функции:

- *Router Discovery* — позволяет узлам локальной сети обнаруживать маршрутизаторы и получать от них сетевые параметры, необходимые для автоконфигурации;
- *Parameter Discovery* — позволяет узлам получать параметры локальной сети и/или маршрутизаторов, например MTU локального канала связи;
- *Prefix Discovery* — используется для определения префикса сети;

- *Address Autoconfiguration* — необходима для автоконфигурации узлов и взаимодействия между ними.

Другая группа функций обеспечивает взаимодействие между узлами:

- *Address Resolution* — функция разрешение IPv6-адресов канального уровня;
- *Next-Hop Determination* — позволяет определить IPv6-адрес назначения пакета и путь до следующего маршрутизатора;
- *Neighbor Unreachability Detection* — позволяет отслеживать состояние каналов связи между соседними узлами локальной сети;
- *Duplicate Address Detection* — позволяет определить дублирование адресов узлов локальной сети.

Последняя группа функций — *Redirect* — используется маршрутизаторами для уведомления узлов о наилучшем маршруте к пункту назначения.

Большинство функций протокола NDP выполняются с использованием пяти сообщений протокола ICMPv6:

1. *Router Solicitation* — отправляется узлами, чтобы запросить любой локальный маршрутизатор отправить сообщение Router Advertisement, не дождаясь следующего периодического объявления. Используется при автоконфигурации узла;

2. *Router Advertisement* — регулярно отправляется маршрутизаторами, чтобы объявить о своем существовании в сети и предоставить узлам информацию о префиксе и/или дополнительных параметрах. Это сообщение также может быть отправлено в ответ на сообщение Router Solicitation;

3. *Neighbor Solicitation* — отправляется узлом, чтобы определить адрес канального уровня соседнего устройства или проверить доступность соседа с помощью адреса канального уровня, хранимого в NDP-таблице. Также используется для определения дублирования адресов (*Duplicate Address Detection*);

4. *Neighbor Advertisement* — отправляется в ответ на сообщение Neighbor Solicitation. Это сообщение также может быть отправлено узлом при изменении адреса канального уровня;

5. *Redirect* — используется маршрутизирующими устройствами для уведомления узлов о наилучшем маршруте к пункту назначения.

6.1. Разрешение адресов IPv6 и определение недоступности соседа

Базовая концепция разрешения IPv6-адресов осталась такой же, как и в IPv4. При необходимости отправки IPv6-пакета соседнему устройству в локальной сети и отсутствии информации о физическом адресе получателя устройство инициирует процесс разрешения адресов. Оно создает в NDP таблице (*Neighbor Cache*) запись, первоначальное состояние которой Incomplete и отправляет соседу сообщение Neighbor Solicitation на групповой адрес Solicited-Node.

При получении сообщения Neighbor Solicitation, устройство назначения должно создать или обновить в NDP-таблице запись, связывающую IPv6-адрес и MAC-адрес соседнего устройства, от которого это сообщение получено. Состояние созданной или обновленной записи устанавливается в Stale.

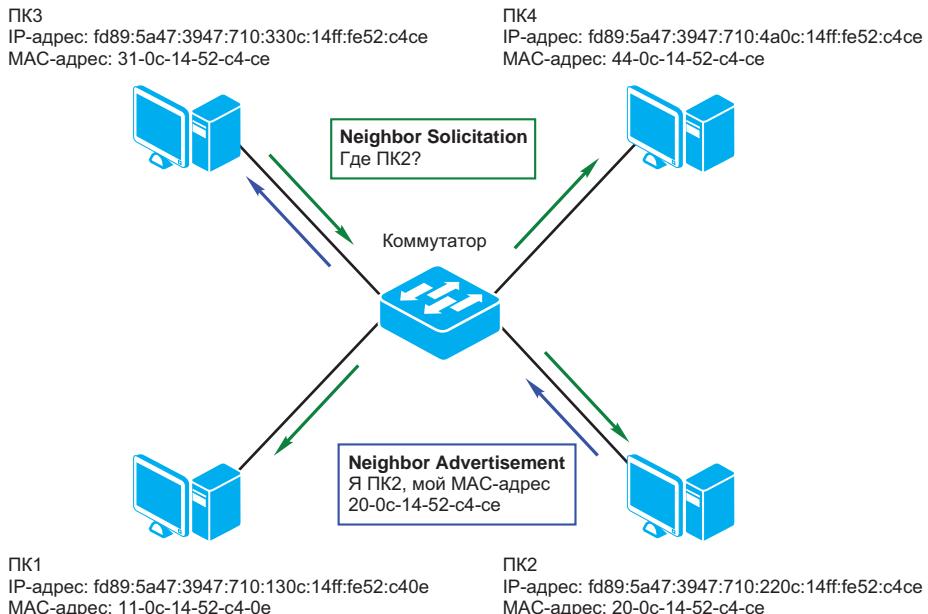


Рис. 6.2. Разрешение адресов с помощью протокола NDP

После этого устройство назначения отправляет в ответ сообщение Neighbor Advertisement. При получении этого сообщения, узел проверяет, была ли соответствующая запись создана в NDP таблице. Если нет, сообщение отбрасывается. Если запись имеется, в нее добавляется полученный MAC-адрес. Состояние записи переходит в Reachable, если в полученном объявлении был установлен флаг Solicited; в противном случае в Stale.

Так же, как и в таблице ARP, в NDP-таблице могут храниться и статические, и динамические записи. На рисунках 6.3 и 6.4 показаны NDP-таблицы коммутатора и подключенного к нему компьютера.

Динамическая запись в NDP-таблице может находиться в одном из пяти состояний:

1. *Incomplete* — состояние, когда сообщение Neighbor Solicitation отправлено на групповой адрес Solicited-Node, но ответное сообщение Neighbor Advertisement еще не получено;

2. *Reachable* — состояние, когда сообщение Neighbor Advertisement получено. Продолжительность этого состояния записи в NDP-таблице ограничена таймером *ReachableTime* (по умолчанию — 30 секунд);

6. Протокол NDP

```
netsh interface ipv6>show neighbors
```

Интерфейс 1: Loopback Pseudo-Interface 1

IP-адрес	Физический адрес	Тип
ff02::2		Постоянный
ff02::c		Постоянный
ff02::16		Постоянный
ff02::1:2		Постоянный
ff02::1:ff00:1		Постоянный
ff02::1:ff00:2		Постоянный
ff02::1:ff23:249a		Постоянный
ff02::1:ffff0:d2d9		Постоянный

Интерфейс 11: Подключение по локальной сети

IP-адрес	Физический адрес	Тип
2001:db8:c1::1	ac-f1-df-b5-fc-00	Устаревший (Маршрутизатор)
ff02::2	33-33-00-00-00-02	Постоянный
ff02::c	33-33-00-00-00-0c	Постоянный
ff02::16	33-33-00-00-00-16	Постоянный
ff02::1:2	33-33-00-01-00-02	Постоянный
ff02::1:3	33-33-00-01-00-03	Постоянный
ff02::1:ff00:1	33-33-ff-00-00-01	Постоянный
ff02::1:ff00:2	33-33-ff-00-00-02	Постоянный
ff02::1:ff12:464	33-33-ff-12-04-64	Постоянный

Рис. 6.3. NDP-таблица компьютера

```
DGS-3120-24TC:admin#show ipv6 neighbor_cache ipif System all
Command: show ipv6 neighbor_cache ipif System all
```

```
2001:DB8:C1::2                               State: Reachable
MAC Address : 00-33-22-33-44-55               Port : 5
Interface   : System                          VID  : 1
```

```
Total Entries: 1
```

Рис. 6.4. NDP-таблица коммутатора

3. *Stale* — состояние, в которое переходит запись по истечении времени таймера ReachableTime с момента последнего получения сообщения Neighbor Advertisement;

4. *Delay* — состояние, в которое переходит запись при передаче данных соседнему устройству. При этом устанавливается таймер *Delay_First_Probe_Time* (по умолчанию — 5 секунд). Если по истечении времени таймера запись все еще остается в состоянии *Delay*, статус записи меняется на *Probe*. Если же подтверждение достижимости было получено, состояние записи меняется на *Reachable*.

5. *Probe* — состояние записи, при котором устройство отправляет сообщение Neighbor Solicitation через промежутки времени, определяемые таймером *RetransTimer* (по умолчанию — 10 секунд). Если в течение трех последовательных передач сообщения Neighbor Solicitation получено сообщение Neighbor Advertisement, то запись переходит в состояние *Reachable*, в противном случае запись удаляется из NDP-таблицы.

Сообщения Neighbor Solicitation и Neighbor Advertisement используются не только для разрешения адресов. У них есть еще одно предназначение — *определение недоступности соседа (Neighbor Unreachability Detection, NUD)*. Функция NUD позволяет отслеживать состояние каналов связи между соседними узлами локальной сети.

Сосед считается доступным, если от него пришло подтверждение, что отправленный ему пакет был успешно принят. Подтверждение может быть получено с помощью протокола верхнего уровня или при получении сообщения Neighbor Advertisement в ответ на отправленное сообщение Neighbor Solicitation.

Каждый раз, когда в результате успешного выполнения разрешения адресов в NDP-таблице создается запись о соседе, ей присваивается статус *Reachable* (доступен) и устанавливается таймер. После истечения таймера, информация о соседе переходит в состояние *Stale* (устарел) и предполагается, что сосед больше не доступен. После того как будет получено подтверждение доступности соседа, состояние записи изменится на *Reachable* и таймер перезапустится.

Операции функции NUD выполняются параллельно с отправкой пакетов соседним устройствам. Если между ними нет обмена данными, то сообщения Neighbor Solicitation и Neighbor Advertisement не отправляются.

6.2. Определение дублирования адресов

При использовании механизма автоконфигурации IPv6-адреса одним из первых шагов этого процесса является определение уникальности сгенерированного адреса Link-Local Unicast для данного сегмента сети. Процесс определения дублирования адресов (*Duplicate Address Detection, DAD*) использует сообщения Neighbor Solicitation и Neighbor Advertisement. Узел отправляет сообщение Neighbor Solicitation, содержащее в качестве адреса

6. Протокол NDP

назначения сгенерированный адрес Link-Local Unicast. Если в ответ на него получено сообщение Neighbor Advertisement, значит, этот адрес уже используется другим узлом.

6.3. Обнаружение маршрутизатора

Одной из важных функций протокола NDP является реализация процесса обнаружения узлами локальных маршрутизаторов (коммутаторов L3) — *Router Discovery*. При этом узлы локальной сети обнаруживают соседние маршрутизаторы (коммутаторы L3) и получают от них сетевые параметры, необходимые для автоконфигурации (рис. 6.5). Операция обнаружения узлами маршрутизаторов выполняется с помощью сообщений ICMPv6 Router Advertisement и Router Solicitation.

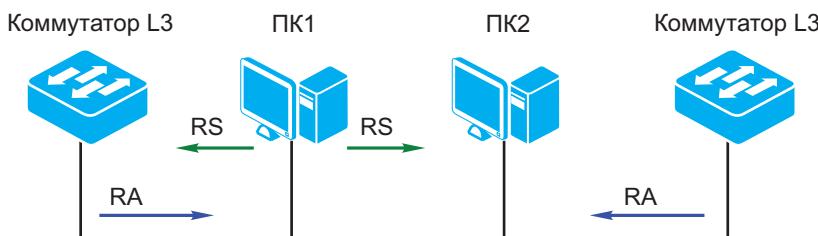


Рис. 6.5. Обнаружение локального коммутатора L3

В процессе обнаружения маршрутизаторов (коммутаторов L3) узлы выполняют следующие функции:

- *Рассылка объявлений*. Узлы прослушивают объявления Router Advertisement, передаваемые маршрутизаторами в локальной сети через определенные интервалы времени и обрабатывают их. Объявления содержат список префиксов, в том числе необходимых для автоконфигурации, а также могут включать информацию о шлюзе по умолчанию;
- *Генерация запросов*. При определенных условиях (например, узел загружается и ему требуются параметры для конфигурации интерфейса) узлы могут генерировать сообщения Router Solicitation. С помощью этого сообщения узел запрашивает любой локальный маршрутизатор о мгновенном предоставлении информации, т. е. об отправке сообщения Router Advertisement;
- *Автоконфигурация*. Если в сети настроен механизм автоконфигурации Stateless autoconfiguration, то узел будет использовать информацию, полученную от локального маршрутизатора, чтобы автоматически сконфигурировать свой IPv6-адрес и другие сетевые параметры.

7. Понятие маршрутизации

Протокол IP является *маршрутизуемым протоколом* (routable protocol), т. е. протоколом, формат пакета которого содержит адресную информацию, позволяющую определять маршрут и доставлять данные между устройствами различных физических сетей, соединенных произвольным образом. Процесс определения пути, по которому IP-пакет будет доставлен адресату, называется *маршрутацией* (routing). Различные физические сети связаны между собой посредством специальных устройств, называемых *маршрутизаторами* (router). Каждый маршрутизатор напрямую подключается как минимум к двум сетям. Основным назначением маршрутизаторов является определение пути следования пакетов и принятие решения об их перенаправлении на одно из ближайших маршрутизирующих устройств.

Обычные компьютеры также участвуют в процессе доставки IP-пакетов. Прежде чем приложение на компьютере-отправителе начнет передачу данных приложению на узле-получателе, необходимо узнать IP-адрес получателя. IP-адрес назначения будет известен сетевому приложению, если его ввел пользователь или он получен в результате разрешения доменных имен с помощью протокола DNS (Domain Name System), например, когда в адресной строке браузера пользователь ввел доменное имя сайта. Далее компьютер должен определить начальный маршрут пакета и решить, какому из ближайших узлов он должен быть переправлен.

Существует два метода, с помощью которых IP-пакет может быть доставлен в пункт назначения: *прямая доставка* (*direct delivery*) и *непрямая доставка* (*indirect delivery*).

Прямая доставка выполняется между двумя узлами, находящимися в одной локальной сети (например, сети Ethernet или Wi-Fi). Узлы могут быть соединены друг с другом с помощью промежуточного устройства, такого как коммутатор или точка доступа. Локальные узлы также могут получать доступ друг к другу и обмениваться информацией без использования каких-либо дополнительных устройств. *Непрямая* доставка происходит в том случае, когда получатель пакета находится в другой локальной сети. При этом отправитель пересыпает пакет ближайшему маршрутизирующему устройству, которое выполняет его дальнейшую доставку конечному получателю.

Подведя итог, можно сделать следующий вывод. Передача IP-пакетов между двумя устройствами, подключенными к одной локальной сети, происходит напрямую без использования маршрутизаторов. Отправитель сообщения помещает пакет в кадр канального уровня, пересыпаемый физическим уровнем непосредственно получателю. Получатель принимает кадр, извлекает из него пакет и передает его на сетевой уровень. Если отправитель не знает MAC-адреса получателя, он может отправить, например, ARP-запрос и, получив ответ, сформировать кадр.

Как отправитель узнает, что он находится с получателем в одной локальной сети? Все просто. IP-адрес состоит из двух частей: номера (префикса)

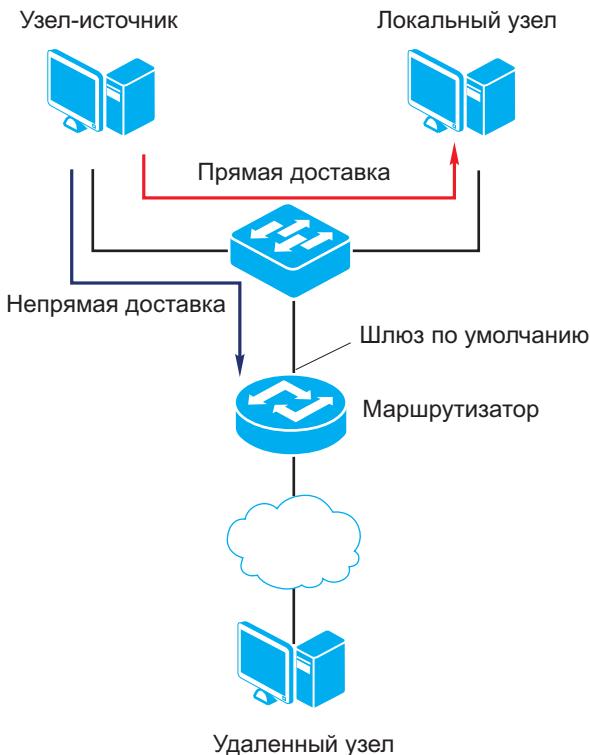


Рис. 7.1. Прямая и непрямая доставки

сети и номера узла. Поэтому, чтобы определить находятся ли отправитель и получатель в одной сети, отправитель должен выделить номер сети из IP-адреса получателя и сравнить его с номером сети, выделенным из его собственного IP-адреса. Если номера сетей совпадают, пакет может быть отправлен получателю напрямую (рис. 7.1).

В составной сети прямая доставка выполняется на завершающем этапе пересылки любого IP-пакета независимо от того, через какое количество сетей и промежуточных устройств он до этого прошел. Последний из маршрутизаторов, находящийся на пути следования пакета и подключенный к одной локальной сети с конечным получателем, выполняет его прямую доставку получателю.

Непрямая доставка — более сложный процесс (см. рис. 7.1). Отправитель не всегда знает, где находится сеть, в которой расположен получатель. Отыскать ее помогают маршрутизирующие устройства. IP-пакет, предназначенный устройству из другой сети/подсети, пересыпается локальному маршрутизатору (коммутатору L3), называемому *шлюзом по умолчанию* (default gateway). Для этого узел локальной сети должен знать IP-адрес шлюза по умолчанию — IP-адрес интерфейса маршрутизатора (коммутатору L3), на

Технологии TCP/IP в современных компьютерных сетях

который перенаправляется весь трафик, не предназначенный для устройств данной локальной сети. Этот адрес указывается в настройках устройств (рис. 7.2). Его можно либо настроить на узле вручную, либо получить динамически. Настройка шлюза по умолчанию не требуется, если передача данных будет выполняться только между устройствами одной локальной сети.

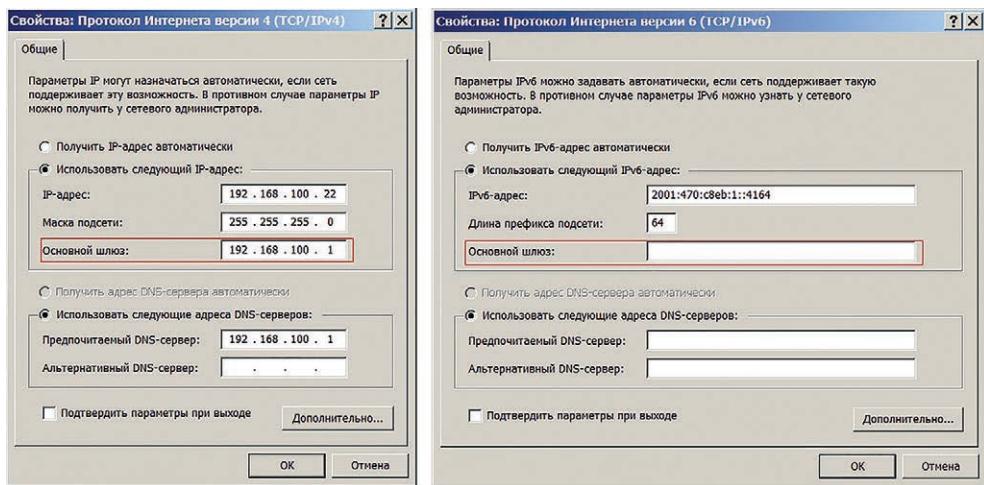


Рис. 7.2. Настройки шлюза по умолчанию в ОС Windows

Поскольку нет возможности напрямую соединить между собой множество компьютеров и неизвестно, где эти устройства находятся, непрямая доставка полагается на промежуточные маршрутизирующие устройства (маршрутизаторы и коммутаторы L3). Как только одному компьютеру надо отправить пакет другому компьютеру, находящемуся в удаленной сети, он инкапсулирует его в кадр и передает по локальной сети шлюзу по умолчанию. Приняв пакет, маршрутизатор анализирует его IP-адрес назначения и определяет следующий шаг (hop) пакета, т. е. ближайший маршрутизатор, которому надо передать пакет, чтобы он был доставлен адресату. Таким образом, пакет передается от одного маршрутизирующего устройства к другому, пока не достигнет маршрутизатора, находящегося с получателем в одной локальной сети. Этот процесс называется *одношаговой маршрутизацией* (next-hop routing).

Процесс маршрутизации пакетов в составных сетях выполняется на основе базы данных маршрутов, называемой *таблицей маршрутизации* (routing table). Она содержит записи, представляющие собой список наилучших маршрутов к определенным сетям и/или узлам. В том случае, если к сети/узлу назначения имеются несколько путей, в таблицу маршрутизации будет помещен маршрут с наилучшей метрикой, т. е. с величиной, определяющей его предпочтительность. Поскольку в процесс маршрутизации вовлечены и компьютеры, и маршрутизирующие устройства, такие таблицы должны

7. Понятие маршрутизации

создаваться на каждом устройстве составной сети независимо от выполняемых им функций. Другими словами, таблица маршрутизации создается не только на маршрутизаторах и коммутаторах L3. Узлы с поддержкой протокола IP также создают таблицу маршрутизации, которая используется для определения наилучшего маршрута для передачи пакета.

В таблице маршрутизации возможны следующие типы записей:

1. *Маршрут к сети*. Маршрут к сети с определенным идентификатором;
2. *Маршрут к узлу*. Маршрут к узлу с определенным сетевым адресом;
3. *Маршрут по умолчанию (default route)*. Маршрут, который используется в том случае, если другой маршрут к пункту назначения неизвестен.

Записи в таблице маршрутизации могут создаваться вручную администратором сети в процессе конфигурации устройства или автоматически в результате работы протоколов динамической маршрутизации.

Записи таблицы маршрутизации обычно содержат следующие поля:

- *адрес назначения (IP Address)* — адрес сети (в некоторых случаях узла назначения);
 - *маска сети (Netmask)* — маска, соответствующая адресу назначения (для сетей IPv4 маска /32 (255.255.255.255) позволяет указать единичный узел сети);
 - *адрес шлюза (Gateway)* — адрес следующего транзитного узла (next hop), т. е. интерфейса ближайшего маршрутизирующего устройства, которому надо отправить пакет, чтобы он достиг сети назначения;
 - *интерфейс (Interface)* — идентификатор интерфейса, который используется узлом для отправки пакета шлюзу;
 - *метрика (Cost)* — числовой показатель, определяющий предпочтительность маршрута. Чем меньше значение метрики, тем более предпочтителен маршрут;
 - *тип протокола (Protocol)* — информация о методе создания записи в таблице маршрутизации.

Количество записей в таблице маршрутизации ограничено техническими характеристиками маршрутизирующего устройства. В технической спецификации указывается поддерживаемое количество записей статической и динамической маршрутизации для протоколов IPv4 (рис. 7.3) и IPv6 (рис. 7.4). В общем случае размер таблицы маршрутизации зависит от количества IP-сетей в автономной системе (рис. 7.5). Количество компьютеров, подключенных к этим IP-сетям, не влияет на размер таблиц маршрутизации, так как в них обычно хранится информация об адресах сетей (а не индивидуальных узлов), к которым подключены потенциальные получатели. Однако как частный случай предусмотрена возможность создания в таблицах маршрутизации маршрутов к индивидуальным узлам. Это позволяет сетевым администраторам более точно распределять потоки данных в сети, тестировать участки сети, а также управлять правами доступа к сетевым ресурсам.

Для того чтобы избавиться от лишней информации в таблице маршрутизации и уменьшить ее размер, можно создать *маршрут по умолчанию (default route)*.

route). Этот маршрут будет использоваться в том случае, если маршрут к требуемому пункту назначения в таблице маршрутизации не найден. Например, чтобы компьютеры выполняли маршрутизацию на основе минимально необходимых данных, в их таблицах маршрутизации создается маршрут по умолчанию, в соответствии с которым все пакеты, предназначенные удаленными узлам, будут пересыпаться на шлюз по умолчанию.

Routing Table

IP Address/Netmask	Gateway	Interface	Cost	Protocol
10.0.0.0/8	0.0.0.0	System	1	Local
172.16.0.0/23	172.16.8.2	iptv-1	2	RIP
172.16.2.0/23	172.16.8.2	iptv-1	2	RIP
172.16.8.0/30	0.0.0.0	iptv-1	1	Local
172.16.8.4/30	0.0.0.0	iptv-2	1	Local
172.16.8.8/30	172.16.8.6	iptv-2	2	RIP
172.16.8.12/30	0.0.0.0	stream	1	Local

Total Entries: 7

Рис. 7.3. Таблица маршрутизации IPv4 коммутатора L3

```
SW2:admin#show ipv6route
Command: show ipv6route

IPv6 Prefix: FDD0:1:2::/64          Protocol: Local    Metric: 1
Next Hop   : ::                      IPIF      : IPIF2

IPv6 Prefix: FDD0:1:3::/64          Protocol: Local    Metric: 1
Next Hop   : ::                      IPIF      : IPIF3

IPv6 Prefix: FDD0:1:4::/64          Protocol: RIPng    Metric: 2
Next Hop   : FE80::86C9:B2FF:FE1F:9C01 IPIF      : IPIF3

IPv6 Prefix: FDD0:1:5::/64          Protocol: Local    Metric: 1
Next Hop   : ::                      IPIF      : IPIF5

IPv6 Prefix: FDD0:1:6::/64          Protocol: RIPng    Metric: 2
Next Hop   : FE80::86C9:B2FF:FE1F:9C01 IPIF      : IPIF3

Total Entries: 5
```

Рис. 7.4. Таблица маршрутизации IPv6 коммутатора L3

Таблица маршрутизации узла

На узле под управлением ОС Windows для отображения таблицы маршрутизации можно использовать команду `route print` или `netstat -r`. Результат работы обеих команд одинаков. С их помощью можно получить следующую информацию:

- **Список интерфейсов:** содержит MAC-адреса сетевых адаптеров, установленных на узле, и присвоенные им номера интерфейсов;

```
C:\>route print
=====
Список интерфейсов
 15...cc 52 af 0a 9c e1 .....Microsoft Virtual WiFi Miniport Adapter
 13...cc 52 af 0a 9c e1 .....Broadcom 4313 802.11b/g/n
 12...98 4b e1 c1 ee 95 .....Realtek PCIe FE Family Controller
 11...e0 2a 82 d7 de 28 .....Устройства Bluetooth (личной сети)
 1.....Software Loopback Interface 1
=====

IPv4 таблица маршрута
=====
Активные маршруты:
Сетевой адрес      Маска сети      Адрес шлюза      Интерфейс      Метрика
        0.0.0.0          0.0.0.0       192.168.100.1   192.168.100.2    25
        127.0.0.0         255.0.0.0      On-link          127.0.0.1     306
        127.0.0.1         255.255.255.255  On-link          127.0.0.1     306
 127.255.255.255  255.255.255.255  On-link          127.0.0.1     306
        192.168.100.0     255.255.255.0  On-link       192.168.100.2    281
        192.168.100.2     255.255.255.255  On-link       192.168.100.2    281
 192.168.100.255  255.255.255.255  On-link       192.168.100.2    281
        224.0.0.0          240.0.0.0      On-link          127.0.0.1     306
        224.0.0.0          240.0.0.0      On-link       192.168.100.2    281
 255.255.255.255  255.255.255.255  On-link          127.0.0.1     306
 255.255.255.255  255.255.255.255  On-link       192.168.100.2    281
=====

Постоянные маршруты:
Сетевой адрес      Маска      Адрес шлюза      Метрика
        0.0.0.0          0.0.0.0       192.168.100.1  По умолчанию
        0.0.0.0          0.0.0.0       192.168.1.1    По умолчанию
=====

IPv6 таблица маршрута
=====
Активные маршруты:
Метрика  Сетевой адрес          Шлюз
 13      281 ::/0                fe80::1
   1      306 ::1/128             On-link
 13      281 fe80::/64           On-link
 13      281 fe80::f1a8:117d:6b9:74a4/128
                                On-link
```

Рис. 7.5. Таблица маршрутизации узла с поддержкой протокола IP

• **Таблица маршрутизации IPv4:** содержит все известные маршруты IPv4, включая подключения к интерфейсу loopback, маршруты локальной сети и маршруты, используемые по умолчанию;

• **Таблица маршрутизации IPv6:** содержит все известные маршруты IPv6, включая подключения к интерфейсу loopback, маршруты локальной сети и маршруты, используемые по умолчанию.

Следует отметить, что информация о сетях, к которым подключен узел, автоматически добавляется в его таблицу маршрутизации.

Записи таблицы маршрутизации IPv4 включают следующую информацию:

- Маршрут по умолчанию **0.0.0.0**;
- Loopback-адреса **127.0.0.0**;

• Адреса узла и локальной сети. На рис. 7.4 это адреса:

192.168.100.0: адрес локальной сети; представляет все узлы в сети 192.168.100.0;

192.168.100.2: адрес локального узла;

192.168.100.255: широковещательный адрес сети, используемый для рассылки сообщений всем узлам локальной сети;

• Групповые адреса **224.0.0.0**, которые зарезервированы для маршрутизации любого многоадресного IP-пакета посредством интерфейса loopback (127.0.0.1) или IP-адреса узла;

• Ограниченный широковещательный адрес **255.255.255.255** — может использоваться для поиска сервера DHCP, до того как будет определен локальный IP-адрес.

Записи таблицы маршрутизации IPv6 включают следующую информацию:

- Маршрут по умолчанию **::/0**;
- Loopback-адрес **::1/128**;

• Сетевой префикс Link-Local fe80::/64 — представляет все узлы внутри сегмента сети IPv6;

• Link-Local IPv6-адрес локального узла. На рис. 7.4 это будет адрес **fe80::f1a8:117d:6b9:74a4/128**;

• Групповые адреса ff00::/8.

Если на узле настроен локальный или глобальный индивидуальный адрес IPv6, он также будет присутствовать в таблице маршрутизации.

7.1. IP-интерфейсы маршрутизирующих коммутаторов

Коммутаторы L3 имеют некоторые особенности, отличающие их от традиционных маршрутизаторов и коммутаторов L2:

- одновременная поддержка функций маршрутизации и коммутации;
- обязательная поддержка механизма VLAN;
- реализация функций маршрутизации на аппаратном уровне с использованием ASIC.

Использование контроллеров ASIC является главной характеристикой, отличающей коммутаторы L3 от традиционных маршрутизаторов, так как за счет выполнения операций аппаратно повышается производительность

7. Понятие маршрутизации

системы, благодаря чему не возникают накладные расходы, связанные с выборкой и интерпретацией хранимых команд. В связи с этим коммутаторы L3 маршрутизируют пакеты в среднем в 10–100 раз быстрее, чем традиционные маршрутизаторы.

Маршрутизирующие коммутаторы поддерживают несколько типов IP-интерфейсов:

- IP-интерфейс виртуальной локальной сети (VLAN) используется в качестве шлюза по умолчанию для узлов локальной сети, подключенных к коммутатору, обеспечивая начальную точку для процесса маршрутизации;
- IP-интерфейс обратной петли (Loopback) является логическим интерфейсом, используемым для управления коммутатором;
- управляющий IP-интерфейс (System) обеспечивает взаимодействие с IP-приложениями управления коммутатором, такими как HTTP (Web-интерфейс), Telnet, TFTP, SNMP и др.

IP-интерфейсы, назначаемые VLAN

В отличие от традиционного маршрутизатора, который требует, чтобы каждый порт был подключен в отдельную сеть или подсеть, физическому порту коммутатора D-Link невозможно назначить собственный IP-интерфейс.

Когда узлы из одной сети/подсети подключаются к портам коммутатора L3, эти порты группируются в VLAN. Даже если для подключения используется только один порт коммутатора, он все равно должен быть помещен в VLAN. Для VLAN создается IP-интерфейс и ему присваивается IP-адрес из сети/подсети, к которой принадлежат подключенные узлы (рис. 7.6). Порты, включаемые в VLAN, не обязательно должны быть расположены последовательно и могут принадлежать разным коммутаторам. IP-интерфейс, назначенный VLAN, может использоваться в качестве шлюза по умолчанию узла данной сети/подсети.

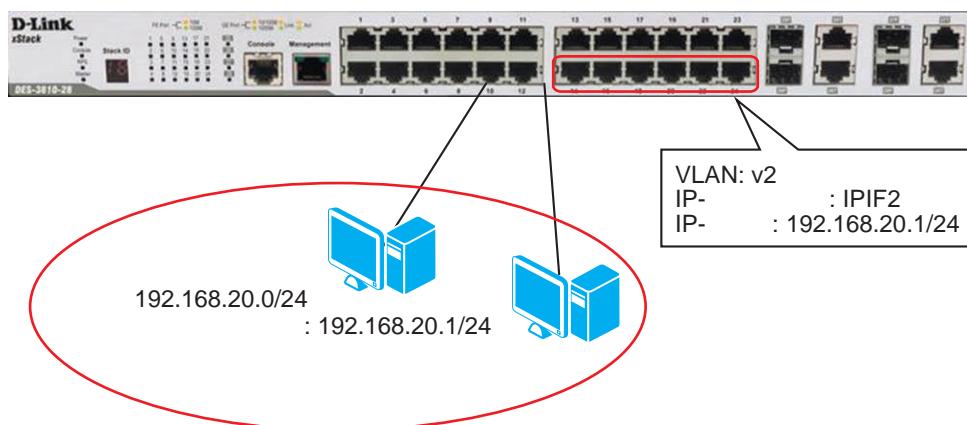


Рис. 7.6. IP-интерфейс, созданный для VLAN

Рассмотрим пример создания IP-интерфейса на коммутаторе L3 (рис. 7.7).

1. Из VLAN по умолчанию (default) удаляются порты 14, 16, 18, 20, 22, 24, к которым подключены узлы сети 192.168.20.0/24, и создается новая VLAN:

```
config vlan default delete 14,16,18,20,22,24  
create vlan v2 tag 2
```

2. Порты 14, 16, 18, 20, 22, 24 добавляются в созданную VLAN v2 как немаркированные порты:

```
config vlan v2 add untagged 14,16,18,20,22,24
```

3. Для VLAN v2 создается IP-интерфейс IPIF2 с адресом 192.168.20.1/24

```
create ipif IPIF2 192.168.20.1/24 v2 state enable
```

Внимание: при создании IP-интерфейса для VLAN, порты которой принадлежат разным коммутаторам, его имя должно быть одинаковым в настройках всех коммутаторов. IP-адреса, назначаемые интерфейсу на разных коммутаторах, должны различаться, но быть из одной сети/подсети.

```
SW1:admin#sh ipif  
Command: show ipif
```

IP Interface	:	IPIF2
VLAN Name	:	v2
Interface Admin State	:	Enabled
IPv4 Address	:	192.168.20.1/24 (Manual) Primary
Proxy ARP	:	Disabled (Local : Disabled)
IP Directed Broadcast	:	Disabled
IPv4 State	:	Enabled
DHCPv6 Client State	:	Disabled
IPv6 State	:	Enabled

Рис. 7.7. Просмотр параметров созданного на коммутаторе L3 IP-интерфейса

Магистральные (Uplink) порты коммутаторов, как правило, могут входить в несколько VLAN и, следовательно, поддерживать несколько IP-интерфейсов (назначенных VLAN). Максимальное количество IP-интерфейсов, которое можно настроить на коммутаторе, ограничено и обычно указывается в технических характеристиках устройства.

Передача данных между узлами одной сети/подсети выполняется на основе анализа MAC-адресов. Другими словами, такой трафик рассматривается как локальный и обрабатывается процессом коммутации. Процессом маршрутизации этот трафик игнорируется.

7. Понятие маршрутизации

Если к коммутатору L3 подключены несколько сетей/подсетей, то для каждой из них настраивается отдельная VLAN, для которой создается IP-интерфейс и которой присваивается IP-адрес.

Когда данные из одной сети передаются в другую сеть, требуется маршрутизация. В пределах одного коммутатора L3 D-Link маршрутизация между VLAN начинает работать сразу после создания IP-интерфейсов. Для объединения сетей, подключенных к разным коммутаторам 3-го уровня, требуется использовать статическую или динамическую маршрутизацию.

IP-интерфейс обратной петли

На коммутаторе можно определить IP-адрес, который не привязан ни к какому порту или VLAN. Он называется интерфейсом обратной петли (Loopback). IP-интерфейс обратной петли является логическим интерфейсом, который не зависит от состояния физических портов и всегда доступен, т. е. он доступен до тех пор, пока пользователь не отключит или не удалит его и пока существует хотя бы одно активное соединение. IP-адрес интерфейса Loopback используется для управления. Также некоторые протоколы маршрутизации, например OSPF, могут использовать его для взаимодействия с соседними маршрутизаторами. IP-адрес интерфейса Loopback не должен использоваться в качестве шлюза по умолчанию и назначаться из адресного пространства IP-адресов, подключенных к коммутатору сетей или подсетей. Количество интерфейсов Loopback, которое можно создать на коммутаторе, ограничено и указано в его технической документации.

IP-интерфейс управления

IP-адрес, который администратор использует при подключении к Web-интерфейсу коммутатора, является адресом его IP-интерфейса управления (System). IP-адрес интерфейса управления обеспечивает поддержку IP-приложений, работающих на коммутаторе, таких как TFTP, SNMP, Telnet, SSL, SSH и др.

В случае использования коммутатора L3 на нем могут быть настроены несколько VLAN и маршрутизация между ними. Маршрутизация требует создания для VLAN IP-интерфейсов. Таким образом, коммутатор L3 будет обрабатывать несколько IP-адресов, и если не настроены соответствующие политики безопасности, для управления им можно использовать любой IP-адрес.

7.2. Архитектура протоколов маршрутизации

Давайте рассмотрим архитектуру протоколов маршрутизации. Под термином «архитектура» в данном случае подразумевается способ структурирования составной сети. Как только появляется большое количество сетей и маршрутизаторов, возникает задача их объединения. Это объединение можно выполнить различными способами. Выбираемая архитектура основана на способе, с помощью которого маршрутизаторы соединяются друг с другом, что в свою очередь влияет на работу протоколов маршрутизации.

Первоначально архитектура ядра Интернета состояла из небольшого количества маршрутизаторов, которые хранили полную информацию о составной сети. По мере развития Интернета стало увеличиваться количество сетей и соединяющих их маршрутизаторов. Это привело к тому, что количество маршрутной информации, обрабатываемой маршрутизаторами, стремительно увеличивалось. Таким образом, возникла необходимость перехода от централизованной к совершенно новой архитектуре, обеспечивающей расширение сети. Новая архитектура рассматривала Интернет как набор независимых групп маршрутизаторов, каждая из которых стала называться автономной системой.

Автономная система (Autonomous system, AS) представляет собой группу маршрутизаторов и IP-сетей, контролируемых одной организацией или находящихся под единым административным управлением, которые используют единую согласованную политику для внутренней маршрутизации (рис. 7.8).

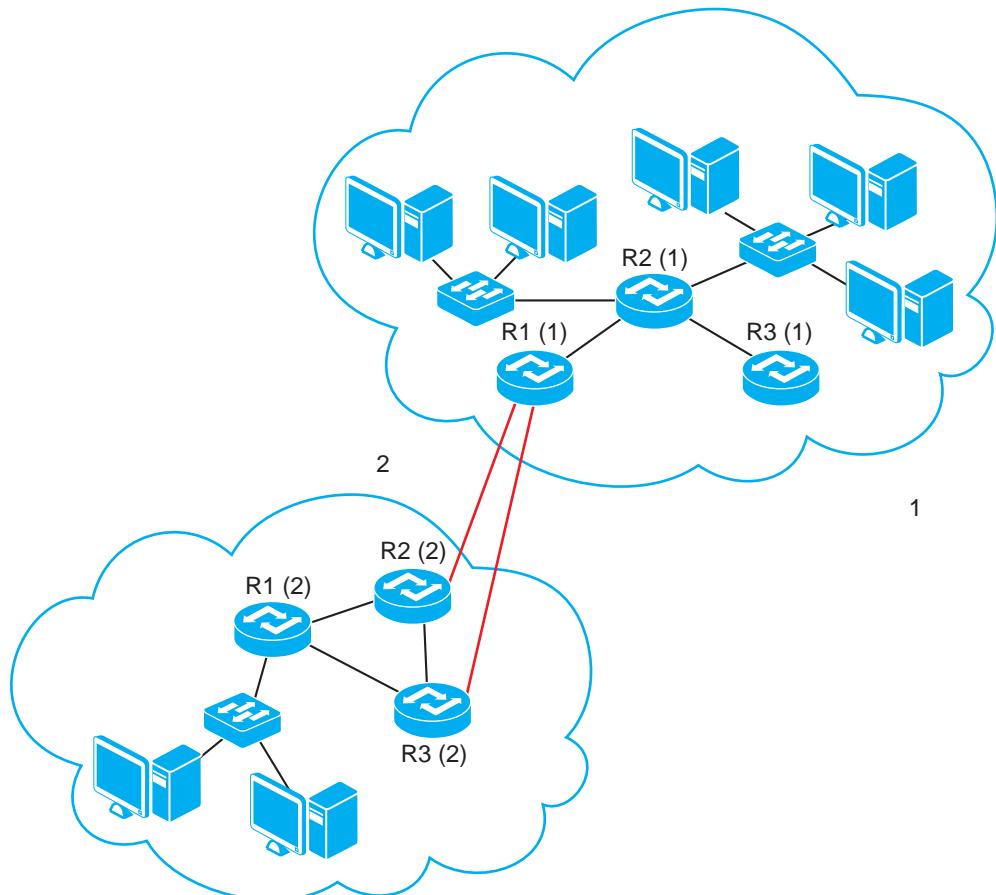


Рис. 7.8. Автономные системы

7. Понятие маршрутизации

Автономная система состоит из маршрутизаторов, которые для внешнего мира (т. е. для других сетей) выглядят как единая сеть. Автономные системы также связываются между собой маршрутизаторами.

Для обмена маршрутной информацией внутри автономной системы используются *внутренние протоколы маршрутизации* (*Interior Routing Protocols*). Каждая AS может использовать отличный от другой AS внутренний протокол маршрутизации. Другими словами, каждая организация сама определяет, какой протокол маршрутизации использовать в своей сети.

Примерами внутренних протоколов маршрутизации являются RIPv1 (Routing Information Protocol version 1), RIPv2 (Routing Information Protocol version 2), RIPng (Routing Information Protocol next generation), OSPF (Open Shortest Path First), OSPFv3 (Open Shortest Path First version 3).

Внешние протоколы маршрутизации (*Exterior Routing Protocols*) используются для обмена маршрутной информацией между автономными системами. Для того чтобы разные автономные системы могли взаимодействовать друг с другом, они должны использовать одинаковые внешние протоколы маршрутизации. Самым известным внешним протоколом маршрутизации является BGP (Border Gateway Protocol).

Набор маршрутизаторов, из которых состоит автономная система, можно разделить на два различных типа:

- *Внутренние маршрутизаторы* (*Internal Routers*) — маршрутизаторы, которые подключаются к маршрутизаторам той же самой автономной системы. На них работают внутренние протоколы маршрутизации;
- *Границочные маршрутизаторы* (*Border Routers*) — маршрутизаторы, устанавливаемые на границе автономной системы. Они подключаются как к маршрутизаторам той же самой автономной системы, так и к маршрутизаторам одной или нескольких других автономных систем. На этих маршрутизаторах работают и внутренние, и внешние протоколы маршрутизации.

Каждая автономная система имеет свой уникальный номер. Агентство IANA выделяет номера автономных систем региональным регистраторам.

7.3. Алгоритмы маршрутизации

Основная функция сетевого уровня заключается в выборе маршрута для передачи пакетов от начальной до конечной точки. Полезно понимать разницу между маршрутизацией, при которой системе приходится делать выбор определенного маршрута следования, и пересылкой — действием, происходящим при получении пакета. Можно представить себе маршрутизатор как устройство, в котором функционируют два процесса. Один из них обрабатывает входящие пакеты и выбирает для них по таблице маршрутизации исходящий интерфейс. Такой процесс называется пересылкой (**forwarding**). Другой процесс отвечает за заполнение и обновление таблиц маршрутизации. Именно здесь начинают работу алгоритмы маршрутизации.

Рассмотрим процесс обработки пакета маршрутизирующим устройством подробнее.

Предположим, что ПК 1, находящийся в сети 192.168.1.0/24, отправляет запрос серверу, расположенному в сети 172.11.10.0/24 (рис. 7.9).

Коммутатор 3-го уровня SW1 получает кадр от ПК 1 на интерфейс Int1 и проверяет его целостность. Если он не поврежден, то у кадра удаляются заголовок и концевик и извлеченный IP-пакет передается на сетевой уровень. В противном случае кадр отбрасывается. На сетевом уровне из заголовка пакета извлекается *IP-адрес назначения* (*Destination address*). Из адреса выделяется номер сети, который сравнивается с записями в таблице маршрутизации. Если совпадений нет и не определен маршрут по умолчанию, то пакет отбрасывается и отправителю посыпается сообщение об ошибке. В нашем примере соответствие найдено. Коммутатор определяет с помощью таблицы исходящий интерфейс (это Int2), и данные передаются на него. Интерфейс Int2 коммутатора SW1 формирует новый кадр, инкапсулируя в него пакет, и пересыпает его следующему на пути коммутатору SW2, адрес которого берется из таблицы (поле *Gateway*).

Следует отметить, что в заголовок передаваемого IP-пакета промежуточными устройствами не вносятся никакие изменения, за исключением уменьшения времени жизни пакета (IPv4) или предельного числа шагов (IPv6). Для IPv4 пересчитывается контрольная сумма заголовка. *IP-адреса в полях заголовка пакета не изменяются*. Они всегда будут указывать на адреса начального отправителя пакета и конечного получателя. В процессе пересылки, если маршрутизатор напрямую не подключен к сети назначения, по таблице маршрутизации определяется адрес следующего промежуточного получателя. В его качестве выступает следующий маршрутизатор или коммутатор L3, расположенный на пути к конечному получателю. Этот IP-адрес никогда не помещается в передаваемый IP-пакет, а служит для формирования кадра на исходящем интерфейсе. На канальном уровне происходит разрешение адресов, т. е. по IP-адресу определяется физический адрес промежуточного устройства, которому надо передать пакет. Исходящий интерфейс инкапсулирует полученный с сетевого уровня пакет в кадр поддерживаемого им протокола. В качестве адреса источника кадра он указывает свой физический адрес, а в качестве адреса назначения — физический адрес непосредственно подключенного к нему интерфейса следующего маршрутизатора. Сформированный кадр отправляется в сеть.

Также надо понимать, что выбор маршрута для каждого передаваемого даже одним источником IP-пакета должен производиться заново, так как оптимальный маршрут мог измениться.

Алгоритмы выбора маршрута можно разбить на два основных класса: неадаптивные и адаптивные. *Неадаптивные алгоритмы* не учитывают при выборе маршрута топологию, текущее состояние сети, загруженность канала, полосу пропускания и т. д. Маршруты создаются вручную администратором сети и автоматически не обновляются. Такая процедура называется *статической маршрутизацией* (static routing). Статическая маршрутизация, как правило, применяется в тех случаях, когда выбор маршрута очевиден.

7. Понятие маршрутизации

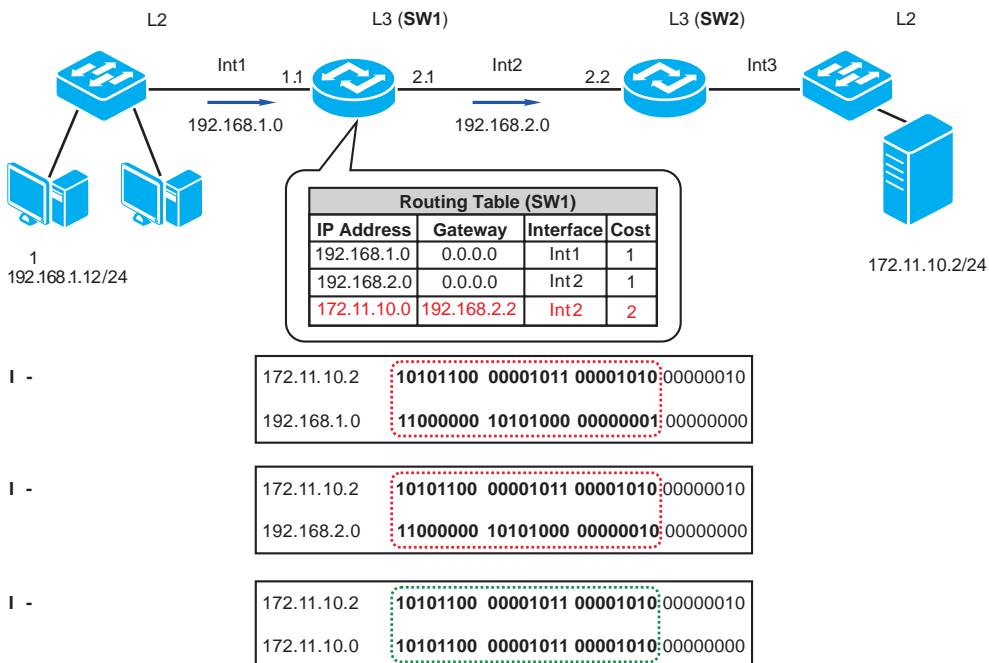


Рис. 7.9. Процесс пересылки пакета маршрутизирующим устройством

Например, ее полезно использовать, когда к сети назначения имеется небольшое количество маршрутов и администратор вручную может задать наилучший маршрут, тем самым снижая нагрузку на процессор маршрутизатора. Также статическую маршрутизацию используют в том случае, если требуется скрыть маршрутную информацию, поскольку при динамической маршрутизации маршрутизирующие устройства обмениваются друг с другом обновлениями о маршрутах, которые могут быть перехвачены злоумышленниками. Если в силу каких-либо причин один из маршрутизаторов, использующих статическую маршрутизацию, выходит из строя, он не сможет оповестить соседей о неисправности, и другие маршрутизаторы будут передавать пакеты по недоступному маршруту. В небольших сетях, где используется два или три маршрутизатора, администратор может решить эту проблему достаточно быстро.

В больших сетях предпочтительнее использовать *динамическую маршрутизацию* (dynamic routing). Алгоритмы динамической маршрутизации (dynamic routing algorithms) являются *адаптивными*, т. е. умеют автоматически создавать записи в таблице маршрутизации и обновлять их при изменении топологии, загруженности линий, обнаружении обрыва соединения и т. д.

Напомним, что в таблицу маршрутизации помещаются лучшие маршруты к пункту назначения. Лучший маршрут выбирается алгоритмом динамической маршрутизации путем сравнения метрик всех маршрутов, найденных

к месту назначения. *Метрика* — это числовой показатель, определяющий предпочтительность маршрута. Чем меньше значение метрики, тем более предпочтителен маршрут. Если маршрутов с одинаковой наименьшей метрикой несколько, они будут помещены в таблицу маршрутизации и использоваться для балансировки нагрузки. При этом не все найденные лучшие маршруты будут помещены в таблицу. Их количество ограничено техническими характеристиками маршрутизирующего устройства.

Протоколы динамической маршрутизации используют разные параметры для вычисления метрики:

- *счетчик промежуточных узлов (Hop count)* или *число переходов* — количество маршрутизаторов (коммутаторов L3), через которые должен пройти пакет, прежде чем достигнет пункта назначения. При прохождении через маршрутизатор (коммутатор L3) значение счетчика узлов увеличивается на 1;

- *задержка передачи (Delay)* — время, требуемое на передачу пакета от отправителя к получателю;

- *надежность линии связи (Reliability)* — обычно обозначает относительное значение количества ошибок для каждого из каналов связи;

- *загруженность (Load)* — средняя загруженность канала связи;

- *полоса пропускания (Bandwidth)* — полоса пропускания канала связи;

- *стоимость (Cost)* — значение, вычисляемое обычно на основе пропускной способности, денежной стоимости или других единиц измерения, называемых администратором.

При этом одни протоколы маршрутизации при расчете метрики используют единственный параметр. Например, RIP использует только число переходов. Другие протоколы могут использовать несколько параметров.

В зависимости от используемого алгоритма протоколы динамической маршрутизации подразделяются на три класса:

- *Дистанционно-векторные протоколы (Distance Vector Protocol)*, основанные на алгоритме Distance Vector Algorithm (DVA), который также называют алгоритмом Белмана — Форда (Bellman — Ford) — маршрутизация по вектору расстояния, которая вычисляет оптимальный маршрут путем обмена списком расстояний между сетями;

- *Протоколы с учетом состояния канала (Link State Protocol)*, основанные на алгоритме Shortest Path First (SPF) — выполняют оценку состояния каналов связи, воссоздают точную топологию сети;

- *Гибридные протоколы* маршрутизации, имеющие черты протоколов как одного, так и другого класса.

Протоколы динамической маршрутизации отличаются методами вычисления маршрутов и обмена маршрутной информацией, максимальным размером сети, временем сходимости и надежностью.

Понятие сходимости

При изменении топологии сети может быть нарушена маршрутизация. Для того чтобы маршрутизация не прерывалась, маршрутизаторы должны владеть непротиворечивыми знаниями о топологии сети.

7. Понятие маршрутизации

Под **сходимостью** (*convergence*) подразумевают то, что все маршрутизаторы работают с одной и той же информацией о местонахождении сетей и о лучших маршрутах к ним. Время, требуемое всем маршрутизаторам для обеспечения сходимости, зависит от расстояния между сетями, используемого протокола маршрутизации, его временных настроек и методов объявления об аварийных ситуациях.

Для алгоритма маршрутизации желательно обладать свойством быстрой сходимости. Это уменьшает время, когда маршрутизаторы используют для принятия решений о выборе маршрута устаревшие знания, из-за чего эти решения могут быть неправильными.

Термины «маршрутизуемый протокол» и «протокол маршрутизации»

Схожее звучание терминов «маршрутизуемый протокол» и «маршрутизирующий протокол» или «протокол маршрутизации» нередко приводит к путанице. Следующие определения помогут понять разницу между этими терминами.

Маршрутизуемый протокол (Routed Protocol) — любой сетевой протокол, который определяет в формате пакета поля, содержащие адресную информацию сетевого уровня, позволяющую осуществлять маршрутизацию. Примерами являются протоколы IPv4/v6 и IPX.

Протокол маршрутизации (Routing Protocol) — протокол, который поддерживает маршрутизуемые протоколы и используется для обмена маршрутной информацией между маршрутизаторами. Сообщения протокола маршрутизации передаются между маршрутизаторами. Примерами являются протоколы RIP, OSPF.

Другими словами, протоколы маршрутизации дают маршрутизаторам возможность выбирать маршрут для маршрутизуемых протоколов, после того как будут обнаружены все возможные пути к получателю.

7.4. Дистанционно-векторные протоколы маршрутизации

Дистанционно-векторный алгоритм маршрутизации (distance vector routing algorithm) иногда называют по именам его создателей алгоритмом Беллмана — Форда (Bellman — Ford). Поиск наилучшего маршрута к сети назначения в них основан на *расстоянии* (distance) между сетями, т. е. на числе переходов или количестве маршрутизаторов между ними. Работают дистанционно-векторные алгоритмы, опираясь на таблицы (т. е. векторы (vector)), поддерживаемые всеми маршрутизаторами и содержащие сведения о кратчайших известных путях к каждому из возможных адресатов и о том, какое соединение следует при этом использовать. Предполагается, что маршрутизаторам известно расстояние до каждого из соседей. Для обновления данных этих таблиц производится обмен информацией с соседними (непосредственно подключенным) маршрутизаторами. Маршрутизаторы периодически рассыпают полные таблицы или их часть своим соседям. Получив таблицу от соседа, маршрутизатор обновляет свою таблицу (вычисляет и заносит

в нее лучший маршрут к сети назначения) и передает ее соседям. Таким образом, шаг за шагом информация о расстоянии распространяется по составной сети.

Этот алгоритм впервые был применен в сети ARPANET в 1969 г. Однако его история тесно связана с разработкой Ethernet. Параллельно с разработкой Ethernet PARC (Xerox's Palo Alto Research Center) создавал высокуровневый протокол, работающий поверх него. Этот протокол получил название Xerox PARC Universal Protocol (PUP). Для обмена маршрутной информацией PUP использовал протокол, называемый Gateway Information Protocol. Позже он был переименован в Routing Information Protocol (RIP) и использовался как часть стека протоколов Xerox Network System (XNS).

RIP получил широкое распространение после того, как был адаптирован для использования в Berkeley Standard Distribution (BSD) операционной системы UNIX. Впервые он появился в BSD версии 4.2 в 1982 году. Чтобы исключить потенциальные проблемы несовместимости, IETF формально определила RIP в RFC 1058 «Routing Information Protocol», опубликованном в июне 1988 года. Первоначальная версия RIP называется RIP версии 1.

В настоящее время существуют три версии протокола:

- RIP версии 1 (RIPv1) используется для поддержки классовой адресации протокола IPv4;
- RIP версии 2 (RIPv2) используется для поддержки бесклассовой адресации IPv4;
- RIPng (next generation) используется для протокола IPv6.

7.4.1. Протокол RIP

Протокол RIP (*Routing Information Protocol*) является представителем класса внутренних протоколов маршрутизации стека TCP/IP. Этот протокол используется в небольших однородных сетях, т. е. имеющих одинаковые характеристики каналов связи, где самый длинный путь между любыми сетями составляет максимум 15 переходов.

Протокол RIP основан на дистанционно-векторном алгоритме маршрутизации и в качестве метрики при выборе маршрута использует *количество переходов (hops count)*, т. е. количество маршрутизаторов, которое должен пройти пакет, прежде чем достичь места назначения. RIP не учитывает ситуации, когда маршрут должен быть выбран на основе таких параметров, как загруженность канала, надежность или задержка передачи.

Каждый маршрутизатор, использующий RIP, хранит таблицу, содержащую записи для каждого пункта назначения (сети или узла) в системе. Каждая запись включает следующую информацию:

- IP address: IP-адрес узла или сети назначения;
- IGateway: адрес первого транзитного маршрутизатора на пути к пункту назначения;
- Interface: интерфейс, напрямую подключенный к первому транзитному маршрутизатору;

7. Понятие маршрутизации

- ICost: числовое значение, показывающее расстояние к пункту назначения;
- ITimer: количество времени, прошедшее после последнего обновления записи.

Если маршрутизатор непосредственно подключен к сети, то расстояние до нее (количество переходов) равно 1. Максимальное число переходов — 15. Значение 16 называется *бесконечностью* (*infinity*) и означает, что данный узел или сеть недостижима.

Периодически (по умолчанию каждые 30 секунд) маршрутизаторы широковещательно отправляют своим соседям обновления с маршрутной информацией. *Обновления* (*update message*) — это набор сообщений, которые содержат часть или всю информацию из таблицы маршрутизации.

При получении обновления от соседа маршрутизатор увеличивает значение метрики (число переходов) к соответствующей сети на 1. Другими словами, если маршрутизатор A отправляет обновление, в котором сообщает, что может достичь сеть X за N шагов, другие маршрутизаторы, непосредственно подключенные к нему, могут достичь сеть X за $N+1$ шагов.

Маршрутизатор, получивший обновление, извлекает из него новую информацию и заносит ее в таблицу маршрутизации. Новой считается информация о маршруте к неизвестной ранее сети или о маршруте к уже известной сети с лучшим значением метрики, т. е. с меньшим расстоянием до нее. Информация об уже известном маршруте с худшим значением метрики (по сравнению с известным маршрутизатору значением) игнорируется.

Для каждой записи в таблице маршрутизации существует *время старения* (*Timeout timer*), контролируемое таймером (по умолчанию 180 секунд). Таймер старения обнуляется каждый раз, когда маршрутизатор получает обновление с информацией о соответствующем маршруте. Если информация о каком-либо маршруте отсутствует в периодических обновлениях и время, установленное таймером, истекает, то маршрут помечается как недостижимый (значение метрики устанавливается равным 16).

Когда маршрут помечается как недостижимый, запускается таймер «*сборщик мусора*» (*Garbage-Collection timer*). Этот таймер отсчитывает время, по истечении которого недостижимый маршрут полностью удаляется из таблицы маршрутизации (по умолчанию 120 секунд). Значение таймеров по умолчанию, используемых протоколом RIP, приведено на рис. 7.10.

В RIPv1 определены два типа сообщений:

- *Request* (*запрос*) — сообщение, отправляемое маршрутизатору с просьбой прислать часть или всю таблицу маршрутизации;
- *Response* (*ответ*) — сообщение, содержащее часть или всю таблицу маршрутизации. Используется для ответа на сообщение Request, отправки периодических обновлений и триггерных обновлений.

Формат сообщения RIPv1 состоит из заголовка и записей о маршрутах (рис. 7.11). Максимальное количество записей равно 25.

```

SW2:admin#show rip
Command: show rip

RIP Global State          : Enabled
Update Time               : 30 seconds
Timeout Time              : 180 seconds
Garbage Collection Time   : 120 seconds

```

Рис. 7.10. Значение таймеров, используемых протоколом RIP

Заголовок содержит следующие поля:

- *Команда (Command)*: значение равно 1 — запрос на получение частичной или полной таблицы маршрутизации; значение равно 2 — ответ, содержащий полную или частичную информацию из таблицы маршрутизации отправителя;
 - *Версия (Version)* — равна 1 для RIPv1.
- Каждая запись о маршруте включает поля:
- *Идентификатор типа адреса (Address Family Identifier)* — тип протокола, используемого в соответствующей сети. Для протокола IP значение равно 2;
 - *IP-адрес (IP Address)* — IP-адрес сети назначения;
 - *Метрика (Metric)* — расстояние до сети (число переходов), указанной в поле *IP-адрес*.

Сообщения RIP инкапсулируются в дейтаграммы UDP. Порт 520 используется в качестве порта источника и приемника. Максимальный размер сообщения 512 байт.

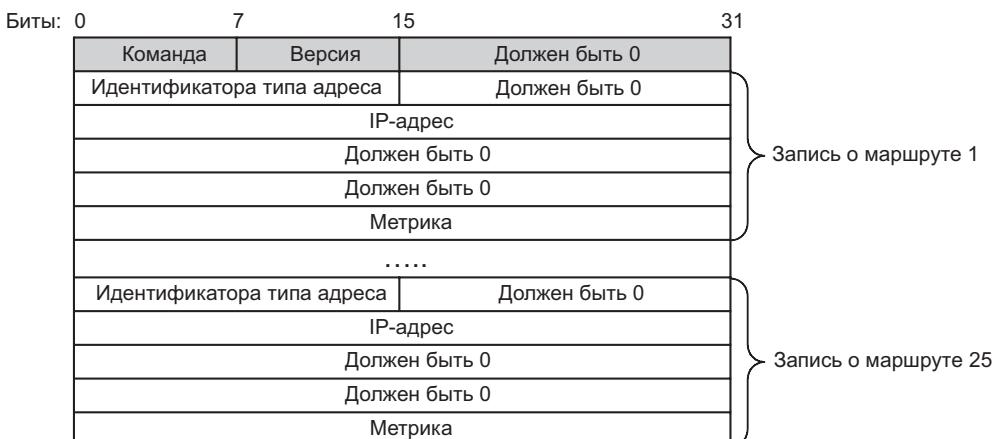


Рис. 7.11. Формат сообщения RIPv1

7. Понятие маршрутизации

Протокол RIPv1 позволяет использовать только классовую (classfull) маршрутизацию, поскольку не включает в маршрутные обновления информацию о маске подсети.

Операции протокола RIP

Работа маршрутизатора (коммутатора L3), использующего протокол RIP, начинается с исследования своих соседей — непосредственно подключенных маршрутизирующих устройств. Первоначальная информация, которая содержится в таблице маршрутизации каждого маршрутизатора RIP, — это информация о непосредственно подключенных к нему сетях, расстояние (Cost) до которых равно 1 (рис. 7.12, момент времени t_0).

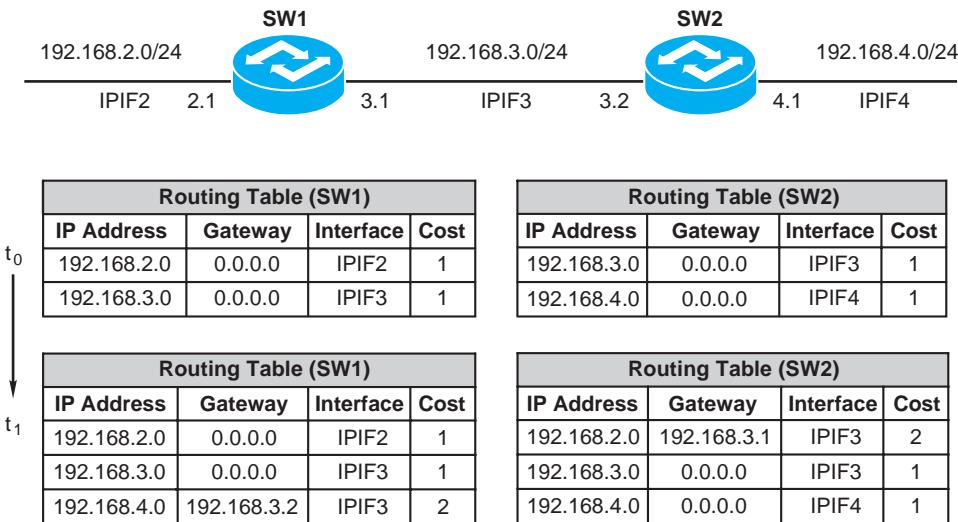


Рис. 7.12. Пример работы протокола RIP

Сразу после инициализации протокола RIP маршрутизатор отправляет широковещательные запросы в непосредственно подключенные к нему сети, чтобы запросить полные таблицы маршрутизации у своих соседей.

При получении RIP-запроса маршрутизатор обрабатывает его и отправляет RIP-ответ, содержащий полную на данный момент времени таблицу маршрутизации.

При получении ответа RIP маршрутизатор сравнивает полученную маршрутную информацию с информацией, хранимой в его таблице маршрутизации. В таблицу маршрутизации добавляются только маршруты к новым или уже известным сетям, но с лучшей метрикой. Обновляя свою таблицу, каждый маршрутизатор увеличивает значение метрики на 1 для каждого добавляемого маршрута (рис. 7.12, момент времени t_1). Эта мера показывает,

Технологии TCP/IP в современных компьютерных сетях

No.	Time	Source	Destination	Protocol	Length	Info
26	19.500351000	192.168.3.2	192.168.3.255	RIPv1	66	Request
27	27.611396000	192.168.2.1	192.168.2.255	RIPv1	66	Response
28	27.612070000	192.168.2.1	192.168.2.255	RIPv1	66	Request
29	27.612757000	192.168.3.1	192.168.3.255	RIPv1	66	Response
30	27.613417000	192.168.3.1	192.168.3.255	RIPv1	66	Request
31	27.615851000	192.168.3.2	192.168.3.1	RIPv1	66	Response

☐ Frame 30: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
 ☐ Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 ☐ Internet Protocol Version 4, Src: 192.168.3.1 (192.168.3.1), Dst: 192.168.3.255 (192.168.3.255)
 ☐ User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
 ☐ Routing Information Protocol
 Command: Request (1)
 Version: RIPv1 (1)
 ☐ Address not specified, Metric: 16
 Address Family: Unspecified (0)
 Metric: 16

Рис. 7.13. Запрос RIP

насколько далеко от маршрутизатора находится сеть назначения в данном направлении.

Чтобы лучше понять принцип работы протокола RIP, вернемся к рис. 7.12 и рассмотрим процесс обработки коммутатором 3-го уровня SW1 полученного обновления и занесения в таблицу маршрутизации новой записи (рис. 7.14).

Шаг 1. Коммутатор SW1 будет игнорировать в полученном обновлении информацию о маршруте к сети 192.168.3.0/24, так как он подключен к ней напрямую и информация о ней уже существует в его таблице маршрутизации с тем же самым значением метрики.

Шаг 2. В таблице маршрутизации SW1 отсутствует маршрут к сети 192.168.4.0/24, поэтому он создает в ней запись для этого маршрута. Эта сеть напрямую подключена к интерфейсу IPIF4 коммутатора 3-го уровня SW2.

No.	Time	Source	Destination	Protocol	Length	Info
26	19.500351000	192.168.3.2	192.168.3.255	RIPv1	66	Request
27	27.611396000	192.168.2.1	192.168.2.255	RIPv1	66	Response
28	27.612070000	192.168.2.1	192.168.2.255	RIPv1	66	Request
29	27.612757000	192.168.3.1	192.168.3.255	RIPv1	66	Response
30	27.613417000	192.168.3.1	192.168.3.255	RIPv1	66	Request
31	27.615851000	192.168.3.2	192.168.3.1	RIPv1	66	Response

☐ Frame 31: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
 ☐ Ethernet II, Src: ac:f1:df:b5:fc:00 (ac:f1:df:b5:fc:00), Dst: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80)
 ☐ Internet Protocol Version 4, Src: 192.168.3.2 (192.168.3.2), Dst: 192.168.3.1 (192.168.3.1)
 ☐ User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
 ☐ Routing Information Protocol
 Command: Response (2)
 Version: RIPv1 (1)
 ☐ IP Address: 192.168.4.0, Metric: 1
 Address Family: IP (2)
 IP Address: 192.168.4.0 (192.168.4.0)
 Metric: 1

Рис. 7.14. Ответ на RIP-запрос

7. Понятие маршрутизации

Шаг 3. В полученном обновлении значение метрики для маршрута к сети 192.168.4.0/24 равно 1. Добавляя маршрут в свою таблицу маршрутизации, SW1 увеличивает ее значение на 1. Значение метрики становится равным 2 (Cost=2).

Шаг 4. В создаваемой записи SW1 указывает в качестве исходящего свой интерфейс IPIF3, поскольку обновление получено через него.

Шаг 5. В качестве шлюза (следующего транзитного узла) использует адрес 192.168.3.2, поскольку обновление получено от отправителя с этим IP-адресом.

Шаг 6. Для созданной записи инициализируется таймер старения.

Для того чтобы все маршрутизаторы имели одинаковую информацию о сетях, их таблицы маршрутизации должны периодически обновляться. Для этого маршрутизаторы используют специальный таймер — *таймер обновлений* (*Update timer*), по истечении времени которого они широковещательно отправляют соседям RIP-ответ, т. е. обновление, содержащее таблицу маршрутизации. По умолчанию значение таймера обновления равно 30 секунд. Этот процесс гарантирует, что маршрутная информация будет рассыпаться регулярно (рис. 7.15).

Если запись для какого-то маршрута в таблице маршрутизации не обновлялась 180 секунд (время, установленное таймером старения по умолчанию), то маршрут считается устаревшим. Информация о маршруте может отсутствовать в рассылаемых обновлениях по причине отключения какого-то маршрутизатора или недоступности сети. Прежде чем все маршрутизаторы узнают о недоступности какой-то сети, в течение 180 секунд они будут считать маршрут доступным и передавать по нему пакеты. Для быстрого уведомления о недоступности маршрута в протоколе RIP используются специальные механизмы.

No.	Time	Source	Destination	Protocol	Length	Info
39	86.887081000	192.168.2.1	192.168.2.255	RIPv1	86	Response
40	86.887775000	192.168.3.1	192.168.3.255	RIPv1	66	Response
41	108.963575000	192.168.3.2	192.168.3.255	RIPv1	66	Response
42	116.885564000	192.168.2.1	192.168.2.255	RIPv1	86	Response

Frame 39: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.2.1 (192.168.2.1), Dst: 192.168.2.255 (192.168.2.255)
User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
Routing Information Protocol
Command: Response (2)
Version: RIPv1 (1)
IP Address: 192.168.3.0, Metric: 1
Address Family: IP (2)
IP Address: 192.168.3.0 (192.168.3.0)
Metric: 1
IP Address: 192.168.4.0, Metric: 2
Address Family: IP (2)
IP Address: 192.168.4.0 (192.168.4.0)
Metric: 2

Рис. 7.15. Обновление RIP

7.4.2. Проблемы при функционировании дистанционно-векторного алгоритма маршрутизации

Алгоритм работы протокола RIP достаточно прост, но обладает медленной сходимостью. Ему требуется достаточно большое время, чтобы информация об изменении топологии достигла всех маршрутизаторов. Медленная сходимость становится наиболее заметной при распространении информации о недоступности маршрутов. Недоступность маршрута может быть определена по истечении времени таймера старения, что добавляет 3 минуты (180 секунд) к времени, через которое сходимость может быть достигнута.

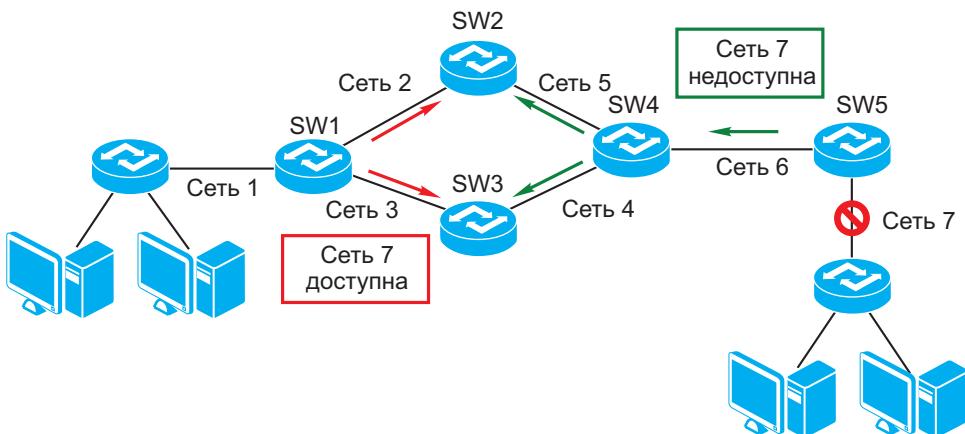


Рис. 7.16. Проблемы при работе дистанционно-векторного алгоритма маршрутизации

В результате медленной сходимости могут возникать *петли маршрутизации*. Петли маршрутизации появляются тогда, когда два или более маршрутизатора пересыпают пакеты по замкнутому пути, вследствие чего они никогда не доходят до нужного получателя. В сетях, где потоки данных значительны, петли маршрутизации могут приводить не только к потере пакетов, но и к неработоспособности всей сети. На рис. 7.16 показана схема, поясняющая эту проблему.

Предположим, что от коммутатора 3-го уровня SW1 наилучший маршрут к сети 7 проходит через коммутатор 3-го уровня SW2 ($\text{Cost}=4$). По какой-то причине в сети 7 произошел разрыв соединения. До этого события все коммутаторы 3-го уровня имели одинаковую информацию о топологии сети. После того как SW5 обнаружил, что сеть 7 недоступна, он отправляет SW4 обновленную маршрутную информацию.

7. Понятие маршрутизации

Коммутатор SW4 обновляет свою таблицу маршрутизации и перестает передавать данные в сеть 7, но SW2, SW3 и SW1 об этом пока не знают, так как еще не получили новую маршрутную информацию.

Во время очередного обновления SW1 отправляет свою таблицу маршрутизации SW2 и SW3, в которой маршрут к сети 7 лежит через коммутатор SW2 ($\text{Cost}=4$). Коммутатор SW3 обновляет свою таблицу маршрутизации, в которой маршрут к сети 7 будет лежать через SW1 ($\text{Cost}=5$). При следующей рассылке маршрутной информации SW3 перешлет обновленную таблицу с неправильным маршрутом коммутатору SW4. Тот, посчитав, что появился альтернативный маршрут к сети 7, обновляет свою таблицу, в которой доступный маршрут будет проходить через SW3 ($\text{Cost}=6$), и при очередной рассылке обновлений перешлет таблицу SW2 и SW5. Таким образом, теперь любой пакет, предназначенный сети 7, будет передаваться по кругу между коммутаторами SW1-SW2-SW4-SW3-SW1, т. е. появится петля маршрутизации. Поэтому основная задача любого протокола динамической маршрутизации заключается в том, чтобы как можно скорее исключить кольцевые маршруты из топологии.

Для решения этой проблемы протокол RIP использует следующие механизмы:

- ограничение максимального числа переходов;
- метод расщепления горизонта (Split Horizon);
- метод расщепления горизонта с испорченным обратным маршрутом (Split Horizon with Poison reverse);
- установка таймеров удержания (Hold down timer);
- триггерные обновления (Triggered Update).

Ограничение максимального числа переходов

Для того чтобы сообщить о недоступности маршрута, дистанционно-векторные протоколы рассылают маршрутную информацию с максимальной метрикой, которая называется *метрикой бесконечности (infinity)*.

Внимание: в любом дистанционно-векторном протоколе определено специальное значение метрики, сообщающее о недоступности сети назначения. Для протокола RIP максимальное значение метрики равно 15. Значение метрики большее 15 называется «infinity» (бесконечность). В RIP в качестве метрики бесконечности используется значение 16.

На рис. 7.17 показан следующий процесс: в сети 192.168.3.0/24, подключенной к коммутатору 3-го уровня SW2, произошел разрыв соединения. Коммутатор SW2 присваивает маршруту максимальное значение метрики ($\text{Cost}=16$) и во время очередного обновления отправляет таблицу коммутатору 3-го уровня SW1, который получает обновленную информацию и хранит маршрут с недостижимой метрикой до тех пор, пока не истечет время таймера Garbage-Collection и маршрут не будет удален из таблицы маршрутизации (рис. 7.18).

Технологии TCP/IP в современных компьютерных сетях

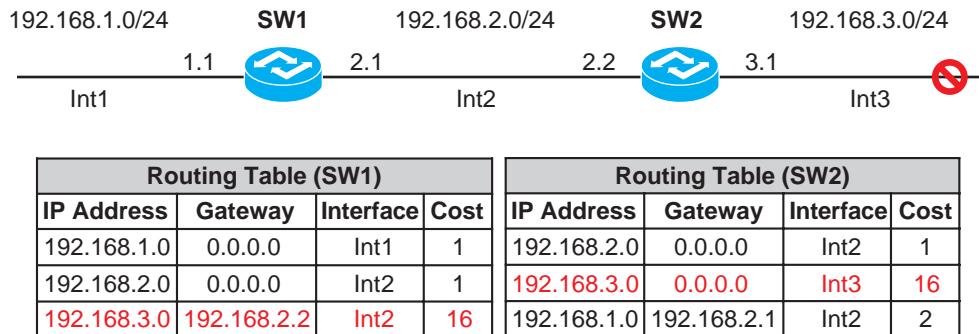


Рис. 7.17. Механизм ограничения максимального числа переходов

```

Frame 44: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: f1:df:b5:fc:00 (ac:f1:df:b5:fc:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.3.2 (192.168.3.2), Dst: 192.168.3.255 (192.168.3.255)
User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
Routing Information Protocol
    Command: Response (2)
    Version: RIPV1 (1)
    IP Address: 192.168.4.0, Metric: 16
        Address Family: IP (2)
        IP Address: 192.168.4.0 (192.168.4.0)
        Metric: 16

```

Рис. 7.18. Обновление с информацией о недоступности сети

Метод расщепления горизонта

Метод *расщепления горизонта* (*Split Horizon*) заключается в том, что информация о маршруте никогда не передается тому маршрутизатору, от которого она была получена (рис. 7.19). Другими словами, когда маршрутизатор отправляет обновление в сеть, он опускает в нем любую информацию о маршрутах, полученных из этой сети. Метод расщепления горизонта не всегда помогает решить проблему петель маршрутизации, особенно в случае, когда несколько маршрутизирующих устройств подключены не напрямую.

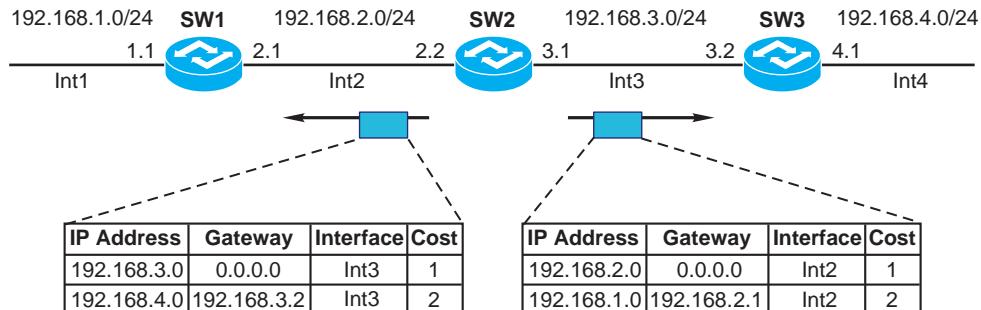
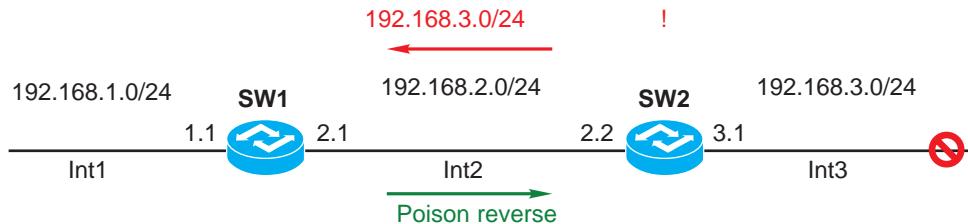


Рис. 7.19. Метод расщепления горизонта

7. Понятие маршрутизации

Метод расщепления горизонта с испорченным обратным маршрутом

Испорченный обратный маршрут (Position reverse) является усовершенствованием метода расщепления горизонта (рис. 7.20). Если на маршрутизатор приходит информация о том, что маршрут до какой-то сети недоступен, этот маршрут не опускается, а включается в обновление, но с метрикой бесконечности (infinity). Другими словами, маршрут «портился».

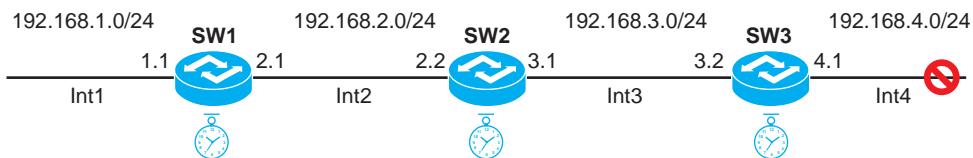


Routing Table (SW1)				Routing Table (SW2)			
IP Address	Gateway	Interface	Cost	IP Address	Gateway	Interface	Cost
192.168.1.0	0.0.0.0	Int1	1	192.168.2.0	0.0.0.0	Int2	1
192.168.2.0	0.0.0.0	Int2	1	192.168.3.0	0.0.0.0	Int3	16
192.168.3.0	192.168.2.2	Int2	16	192.168.1.0	192.168.2.1	Int2	2

Рис. 7.20. Метод расщепления горизонта с испорченным обратным маршрутом

Установка таймера удержания

Механизм установки таймера удержания (*Hold down timer*) позволяет маршрутизатору запускать таймер при получении информации о недоступности сети и игнорировать в течение времени, установленного таймером, все полученные обновления о доступности маршрута. Длительность таймера, определенная в RFC 2091 равна 120 секундам. В течение времени, установленного таймером, маршрут объявляется как недоступный.



Routing Table (SW1)				Routing Table (SW2)				Routing Table (SW3)			
IP Address	Gateway	Interface	Cost	IP Address	Gateway	Interface	Cost	IP Address	Gateway	Interface	Cost
192.168.1.0	0.0.0.0	Int1	1	192.168.2.0	0.0.0.0	Int2	1	192.168.3.0	0.0.0.0	Int3	1
192.168.2.0	0.0.0.0	Int2	1	192.168.3.0	0.0.0.0	Int3	1	192.168.4.0	0.0.0.0	Int4	1
192.168.3.0	192.168.2.2	Int2	2	192.168.1.0	192.168.2.1	Int2	2	192.168.2.0	192.168.3.1	Int3	2
192.168.4.0	192.168.2.2	Int2	3	192.168.4.0	192.168.3.2	Int3	2	192.168.1.0	192.168.3.1	Int3	3

Рис. 7.21. Механизм установки таймера удержания

Рассмотрим принцип работы этого механизма (рис. 7.21). В сети 192.168.4.0/24 произошел обрыв соединения. Коммутатор 3-го уровня SW3 отправляет обновление коммутатору 3-го уровня SW2 о том, что эта сеть недоступна. SW2, получив от SW3 обновление, помечает маршрут как недоступный и запускает таймер удержания. Если в какой-то момент до истечения периода времени, установленного таймером удержания, SW2 получит от того же соседа обновление, которое сообщает, что ранее недоступная сеть теперь доступна, он помечает эту сеть как доступную и сбрасывает таймер удержания. Если до истечения таймера SW2 получит от SW1 обновление о доступности сети с лучшей метрикой по сравнению с той, которая записана в его таблице, он помечает маршрут как доступный и сбрасывает таймер. Если SW2 получает обновление с худшней метрикой, то эта маршрутная информация игнорируется.

Триггерные обновления

Решение проблемы возникновения петель маршрутизации возможно при оперативной рассылке обновленной информации о разрыве соединения, что существенно снижает вероятность зацикливания пакетов. *Метод триггерных обновлений (triggered update)* позволяет маршрутизирующему устройству мгновенно, не дожидаясь очередного цикла обновления маршрутной информации, отправлять соседним маршрутизирующим устройствам информацию об аварийном маршруте. Этот прием может во многих случаях предотвратить передачу устаревших сведений об отказавшем маршруте, но перегружает сеть служебными сообщениями, поэтому триггерные объявления отправляются с некоторой задержкой. Триггерные обновления не позволяют избежать петель маршрутизации без использования дополнительных методов, например метода расщепления горизонта с испорченным обратным маршрутом (Split Horizon with Posion reverse).

7.4.3. Протокол RIPv2

Протокол RIPv2 является расширением RIPv1 и определен в RFC 2453. Принцип работы RIPv2 в основном тот же, что и RIPv1. Главные отличия RIPv2 от RIPv1 состоят в следующем:

- *Поддержка бесклассовой адресации.* Протокол RIPv2 включает в маршрутные обновления информацию о маске подсети для каждого сетевого адреса, тем самым позволяя поддерживать маски переменной длины (VLSM) и бесклассовую маршрутизацию (CIDR);

- *Указание адреса следующего маршрутизатора.* Каждая запись включает IP-адрес маршрутизатора (коммутатора 3-го уровня), который может быть использован в качестве следующего транзитного маршрутизатора, предназначенного для передачи пакета в сеть назначения. Это позволяет повысить эффективность маршрутизации, избежав лишних пересылок;

- *Аутентификация.* Протокол RIPv2 предоставляет базовый механизм аутентификации, который позволяет маршрутизаторам идентифицировать другие маршрутизаторы, прежде чем принимать RIP-сообщения от них;

7. Понятие маршрутизации

• **Метки маршрута (Route Tag).** Каждая запись RIPv2 содержит поле *Route Tag* (метка маршрута), в котором хранится дополнительная информация о маршруте. Это поле используется для идентификации автономной системы при работе с протоколами внешней маршрутизации;

• **Использование многоадресной рассылки.** Для уменьшения нагрузки на сеть протокол RIPv2 позволяет рассыпать маршрутные обновления не широковещательным методом, а на специальный групповой адрес 224.0.0.9.

Формат сообщения протокола RIPv2 в целом аналогичен формату RIPv1 (рис. 7.22). Маршрутизаторы, использующие протокол RIPv1, могут принимать маршрутные обновления RIPv2 (рис. 7.23).

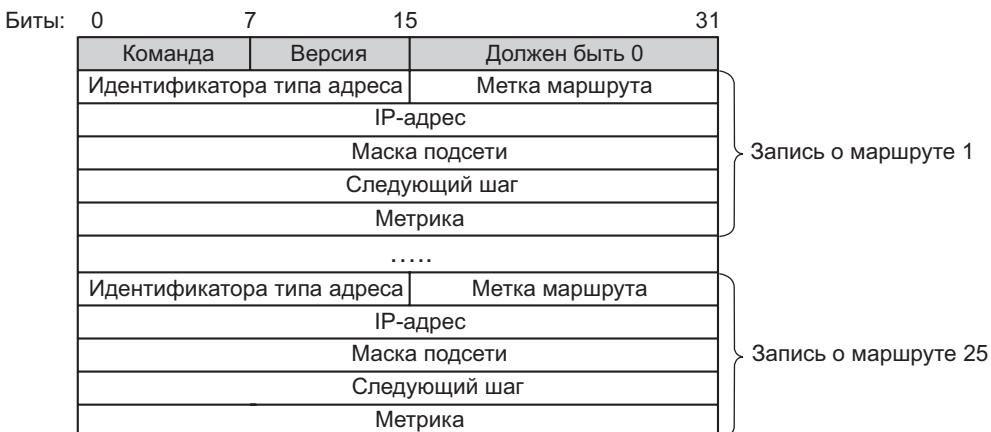


Рис. 7.22. Формат сообщения протокола RIPv2

No.	Time	Source	Destination	Protocol	Length	Info
8	0.773397000	10.90.90.90	224.0.0.9	RIPv2	86	Response
9	0.774243000	192.168.4.2	224.0.0.9	RIPv2	86	Response

Frame 9: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
Ethernet II, Src: 84:c9:b2:1f:9c:03 (84:c9:b2:1f:9c:03), Dst: IPv4mcast_00:00:09 (01:00:5e:00:00:09)
Internet Protocol Version 4, Src: 192.168.4.2 (192.168.4.2), Dst: 224.0.0.9 (224.0.0.9)
User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
Routing Information Protocol
 Command: Response (2)
 Version: RIPv2 (2)
 IP Address: 192.168.2.0, Metric: 2
 Address Family: IP (2)
 Route Tag: 0
 IP Address: 192.168.2.0 (192.168.2.0)
 Netmask: 255.255.255.0 (255.255.255.0)
 Next Hop: 0.0.0.0 (0.0.0.0)
 Metric: 2
 IP Address: 192.168.5.0, Metric: 2
 Address Family: IP (2)
 Route Tag: 0
 IP Address: 192.168.5.0 (192.168.5.0)
 Netmask: 255.255.255.0 (255.255.255.0)
 Next Hop: 0.0.0.0 (0.0.0.0)
 Metric: 2

Рис. 7.23. Обновление RIPv2

В сообщение протокола RIPv2 добавлены следующие новые поля:

- *метка маршрута (Route tag)* — используется для идентификации автономной системы при работе с протоколами внешней маршрутизации;
- *маска подсети (Subnet Mask)*;
- *следующий шаг (Next Hop)* — IP-адрес следующего транзитного маршрутизатора на пути к сети назначения.

7.4.4. Протокол RIPng

Протокол *RIP next generation (RIPng)* представляет собой версию протокола RIP, разработанную для поддержки IPv6, и определен в RFC 2080. Он поддерживает улучшения базовой версии протокола RIP, реализованные в RIPv2, и отличается тем, что использует адреса в формате IPv6.



Рис. 7.24. Формат сообщения протокола RIPng

No.	Time	Source	Destination	Protocol	Length	Info
7	0.083459000	fe80::86c9:b2ff:fe1f:9cff02::9		RIPng	126	Command Response, Version 1
8	0.084284000	fe80::86c9:b2ff:fe1f:9cff02::9		RIPng	126	Command Response, Version 1
Frame 7: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)						
Ethernet II, Src: 84:c9:b2:1f:9c:02 (84:c9:b2:1f:9c:02), Dst: IPv6mcast_00:00:00:00:00:09 (33:33:00:00:00:09)						
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c02 (fe80::86c9:b2ff:fe1f:9c02), Dst: ff02::9 (ff02::9)						
User Datagram Protocol, Src Port: ripng (521), Dst Port: ripng (521)						
RIPng						
Command: Response (2)						
Version: 1						
Reserved: 0000						
Route Table Entry: IPv6 Prefix: 134f::/64 Metric: 1						
IPv6 Prefix: 134f:: (134f::)						
Route Tag: 0x0000						
Prefix Length: 64						
Metric: 1						
Route Table Entry: IPv6 Prefix: 134b::/64 Metric: 1						
IPv6 Prefix: 134b:: (134b::)						
Route Tag: 0x0000						
Prefix Length: 64						
Metric: 1						
Route Table Entry: IPv6 Prefix: 134c::/64 Metric: 1						
IPv6 Prefix: 134c:: (134c::)						
Route Tag: 0x0000						
Prefix Length: 64						
Metric: 1						

Рис. 7.25. Обновление RIPng

Основные особенности RIPng:

- *Поддержка бесклассовой адресации.* Вместо поля *Маска подсети* используется поле *Префикс*;
- *Указание адреса следующего маршрутизатора.* Из-за большого размера IPv6-адреса это поле является дополнительным; если требуется указать адрес транзитного маршрутизатора, он указывается в отдельной записи;
- *Аутентификация.* Протокол RIPng не имеет собственного механизма аутентификации и опирается на стандартную функцию IPSec, обязательно поддерживающую в IPv6;
- *Использование многоадресной рассылки.* Сообщения отправляются на специальный групповой адрес — FF02::9.

Формат сообщения RIPng аналогичен формату RIPv1/v2 за исключением длины записей (рис. 7.24). В отличие от RIPv1/v2 сообщения RIPng передаются, используя UDP-порт 521. Максимальное количество записей сообщения RIPng не ограничено 25, как в RIPv1/v2, а ограничено только *MTU (Maximum Transmission Unit)* сети, через которую будет передаваться сообщение.

7.5. Протокол OSPF

Протоколы RIPv1, RIPv2, RIPng, основанные на дистанционно-векторном алгоритме, хорошо работали в небольших однородных сетях. Однако по мере увеличения размера автономных систем начали проявляться их недостатки, связанные с медленной сходимостью, недостаточной надежностью, ограниченным размером сети, размером пакетов обновлений и т. д.

Недостатки протокола RIP привели к разработке нового протокола маршрутизации. Работа над созданием первого протокола маршрутизации на основе состояния канала началась еще в 1970-х годах для использования в сети ARPANET. В 1988 году IETF создала рабочую группу и начала работу над протоколом, основанным на алгоритме состояния канала (*Link State Algorithm*), также называемом *Shortest Path First (SPF)*. Этот протокол получил название **Open Shortest Path First (OSPF)**. В названии протокола заключены две его важные характеристики: он основан на открытом стандарте, т. е. для его использования не требуется лицензия (*Open*), и позволяет маршрутизаторам определять кратчайший путь между двумя сетями (*SPF*).

Первая версия OSPF была описана в RFC 1131, опубликованном в октябре 1989 года. Эту версию в июле 1991 года заменила OSPF version 2, описанная в RFC 1247. Далее последовали несколько ревизий стандарта OSPF version 2, последней из которых является RFC 2328. Именно OSPF version 2 в настоящее время подразумевается под термином «OSPF».

Основными целями разработки OSPF были уменьшение трафика, создаваемого протоколом маршрутизации, и быстрая сходимость. Эти задачи протокол решил, но при этом оказался очень ресурсоемким. Ему требуется много памяти для хранения различных таблиц, а при вычислении кратчайших путей по алгоритму SPF он сильно нагружает центральный процессор.

7.5.1. Обзор протокола

Протокол OSPF является протоколом динамической маршрутизации. Он способен быстро обнаруживать изменения топологии в автономной системе и вычислять новые маршруты, свободные от петель после окончания периода сходимости. Период сходимости короткий, он включает минимальный обмен трафиком маршрутизации.

В протоколах с учетом состояния канала каждый маршрутизатор обслуживает базу данных, описывающую топологию автономной системы. Эта база данных называется *базой данных состояния канала* (Link State Database, LSDB). Она идентична на каждом маршрутизаторе. Каждая отдельная часть этой базы данных описывает локальное состояние определенного маршрутизатора, т. е. используемые им интерфейсы и его доступных соседей. Все маршрутизаторы рассылают объявления о своем локальном состоянии через автономную систему.

Все маршрутизаторы параллельно выполняют один и тот же алгоритм для вычисления кратчайших маршрутов. Основываясь на информации из базы данных состояния канала, каждый маршрутизатор строит дерево кратчайших маршрутов, указывая себя в качестве его корня. Дерево кратчайших маршрутов определяет маршруты к каждому пункту назначения в автономной системе. Эти маршруты заносятся в таблицу маршрутизации. Если к пункту назначения имеется несколько равных по стоимости маршрутов, трафик равномерно распределяется по ним. Стоимость маршрута описывается единой безразмерной метрикой.

Протокол OSPF позволяет группировать вместе набор непрерывных сетей и маршрутизаторов. Такая группа называется *областью* (area). Другими словами, область представляет собой IP-сеть, разбитую на подсети. OSPF поддерживает VLSM и позволяет гибко настраивать IP-подсети, т. е. подсети одной IP-сети могут иметь разный размер. OSPF передает информацию о подсети в своих обновлениях.

Автономная система OSPF может состоять из одной или нескольких областей (рис. 7.26). Области нумеруются и независимо друг от друга управляются маршрутизаторами, находящимися внутри них. Каждая область имеет свою собственную базу данных состояния канала. Информация о топологии области не распространяется за ее пределы, что позволяет значительно уменьшить трафик маршрутизации.

Если сеть небольшая, то она может управляться как единый объект и иметь только одну область.

Большие сети с десятками или сотнями маршрутизаторов рекомендуется разбивать на несколько областей, так как использование плоской структуры приводит к снижению производительности. Проблема возникает из-за большого количества маршрутной информации, которую требуется передавать и обрабатывать. Необходимость хранения маршрутизаторами больших таблиц маршрутизации и баз данных состояния канала приводит к сильному потреблению памяти. Изменения топологии, которые неизбежны в больших сетях,

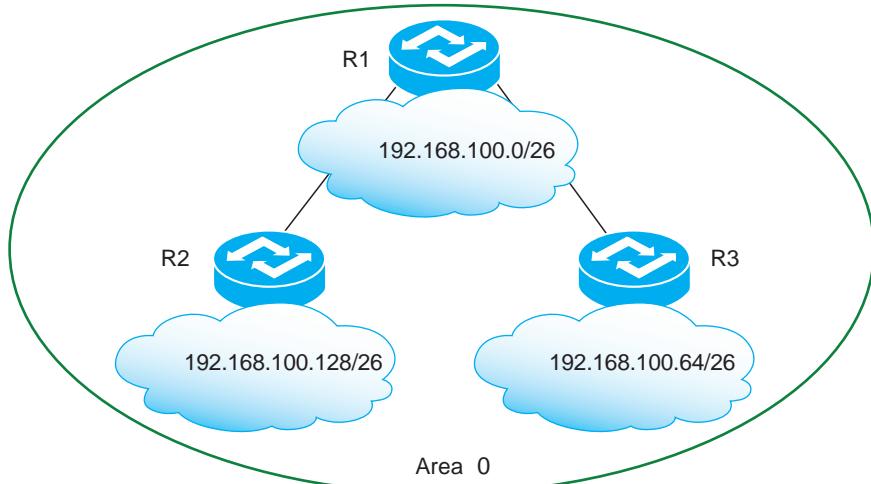


Рис. 7.26. Область OSPF

запускают процесс вычислений по алгоритму SPF и сильно загружают процессоры маршрутизаторов, особенно если это происходит часто. В результате этого устройства перестают нормально справляться со своей работой. Разделение большой составной сети на множество областей позволяет уменьшить количество трафика маршрутизации, размер таблиц маршрутизации, частоту вычислений алгоритма SPF.

Разделение автономной системы на несколько областей называется *иерархической топологией* (рис. 7.27). В ней имеются два уровня. На верхнем уровне находится *магистральная область* (Backbone area), к которой подключаются все остальные области, находящиеся на нижнем уровне. Соединение

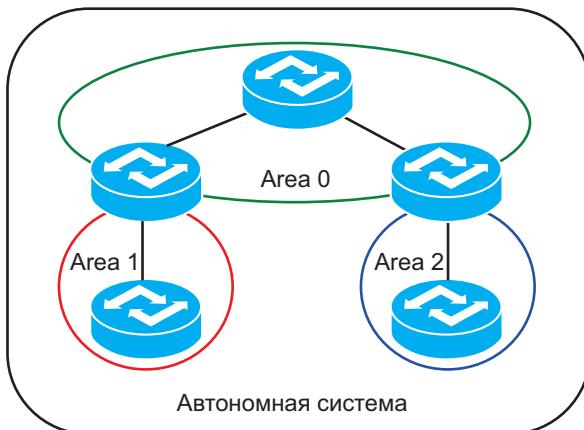


Рис. 7.27. Разделение автономной системы на области

областей через логическую магистральную область позволяет им обмениваться друг с другом маршрутной информацией через всю автономную систему (рис. 7.28).

В иерархической топологии маршрутизаторы играют различные роли в зависимости от того, где и как они подключены. Маршрутизаторы могут быть отнесены к следующим четырем перекрывающимся типам (т. е. они могут принадлежать более чем к одному типу):

- **Внутренний маршрутизатор (Internal router)** — маршрутизатор, непосредственно подключенный к сетям или маршрутизаторам, принадлежащим одной области. Внутренние маршрутизаторы имеют одинаковую базу данных состояния канала и выполняют одну копию основного алгоритма маршрутизации. Другими словами, они обслуживают базу данных состояния канала только для этой области и не знают о топологии других областей;

- **Границочный маршрутизатор области (Area border router, ABR)** — маршрутизатор, который подключен к множеству областей. ABR хранят отдельные базы данных состояния канала для каждой области, к которой они подключены. Они суммируют информацию о топологии подключенных областей для распространения по магистрали. Магистраль в свою очередь передает эту информацию другим областям;

- **Магистральный маршрутизатор (Backbone router)** — маршрутизатор, который имеет интерфейс к магистральной области (Area 0). Он включает все ABR, так как эти маршрутизаторы передают маршрутную информацию между областями. Однако магистральным маршрутизатором также может быть маршрутизатор, подключенный только к магистральной области. Другими словами, ABR всегда является магистральным маршрутизатором, а магистральный маршрутизатор необязательно будет ABR;

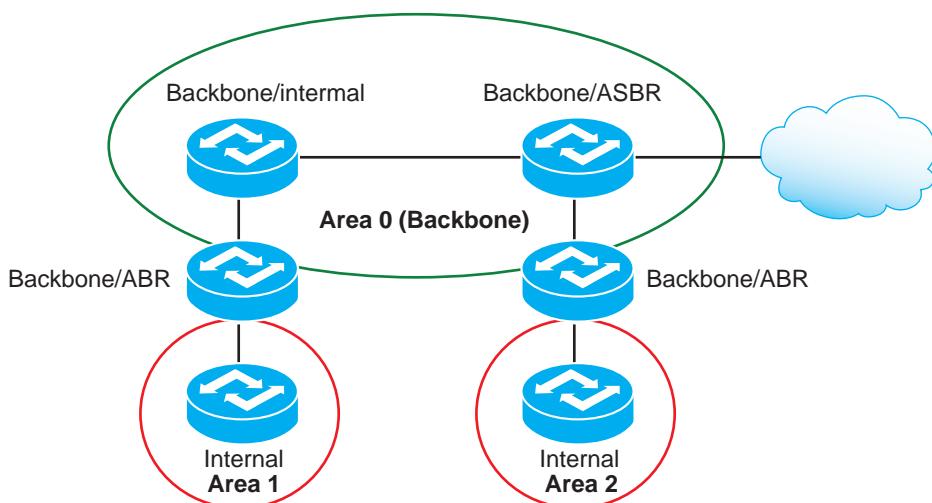


Рис. 7.28. Компоненты сети OSPF с множеством областей

7. Понятие маршрутизации

- **Границный маршрутизатор автономной системы** (*AS boundary router, ASBR*) — маршрутизатор, который обменивается маршрутной информацией с маршрутизаторами, принадлежащими другой автономной системе.

Маршрутизация в автономной системе выполняется на двух уровнях в зависимости от того, где находятся источник и приемник пакета — в одной области (*intra-area routing*) или в разных (*inter-area routing*).

Маршрутизация внутри области определяется ее собственной топологией и выполняется на основе информации, полученной из области. Информация, полученная извне, не используется. Это защищает маршрутизацию внутри области от получения неверной маршрутной информации. Другими словами, если источник и приемник принадлежат одной области, то маршрутизация выполняется только между сетями и маршрутизаторами этой области.

Если источник и приемник в разных областях, пакет маршрутизируется от источника маршрутизатору ABR из его области, далее через магистраль маршрутизатору ABR в области приемника и, наконец, доставляется в пункт назначения.

Все обмены информацией протокола OSPF могут быть аутентифицированы. Это означает, что только надежные маршрутизаторы смогут маршрутизировать трафик. Для аутентификации могут использоваться различные методы.

Внимание: когда конфигурируется сеть, состоящая только из одной области, всегда необходимо использовать Area 0 (Area 0.0.0.0), которая определена как магистральная (*backbone*) область автономной системы.

Определения основных терминов OSPF

Автономная система (*Autonomous System, AS*) — группа маршрутизаторов, обменивающихся маршрутной информацией с использованием общего протокола маршрутизации.

Область (*Area*) — группа непрерывных сетей и маршрутизаторов, которые имеют одинаковый идентификатор области.

Идентификатор области (*Area ID*) — номер, который идентифицирует область. В зависимости от реализации программного обеспечения маршрутизатора и предпочтений пользователя он может быть записан как десятичное число или в точечно-десятичной нотации как IP-адрес. Коммутаторы L3 D-Link поддерживают оба формата.

Магистральная область (*Backbone area*) — специальная область OSPF Area 0 (Area 0.0.0.0), отвечающая за распространение маршрутной информации между немагистральными областями. Другими словами, при соединении множества областей магистральная область является центром, к которому подключены все остальные области.

Внутренний маршрутизатор (*Internal router*) — маршрутизатор, непосредственно подключенный к сетям, принадлежащим одной области.

Идентификатор маршрутизатора (*Router ID*) — 32-битный номер, назначаемый маршрутизатору, выполняющему протокол OSPF. Он уникально определяет маршрутизатор внутри автономной системы.

Канал (*Link*) — интерфейс маршрутизатора, с помощью которого он подключается к сети. Интерфейс имеет связанную с ним информацию о состоянии, которая поступает от протоколов нижележащих уровней или непосредственно от протокола маршрутизации.

Соседние маршрутизаторы (*Neighboring routers*) — два маршрутизатора, которые имеют интерфейсы к общей сети.

Соседство (*Adjacency*) — отношения, установленные между выбранными соседними маршрутизаторами с целью обмена маршрутной информацией. Установление соседства выполняется динамически с помощью протокола OSPF Hello. Не каждая пара соседних маршрутизаторов может установить соседство.

Объявление о состоянии канала (*Link State Advertisement, LSA*) — блок данных, описывающий локальное состояние маршрутизатора или сети. Маршрутизатор рассыпает LSA через домен OSPF-маршрутизации. Существуют несколько типов LSA — для рассылки внутри области, для рассылки за пределы области и рассылки через AS.

Каждый маршрутизатор хранит полученные LSA в **базе данных состояния канала** (*Link State Database*). Маршрутизаторы из одной области имеют одинаковую базу данных состояния канала. Маршрутизаторы из разных областей имеют разные базы данных состояния канала. Если маршрутизатор подключен к разным областям, он хранит отдельные базы данных состояния канала для каждой области.

Типы сетей OSPF

Маршрутизаторы, выполняющие протокол OSPF, могут обмениваться маршрутной информацией только после установления соседства. Поэтому каждый маршрутизатор должен установить соседство как минимум с одним маршрутизатором из каждой сети, к которой он подключен. Количество устройств, с которыми должно быть установлено соседство, зависит от типа сети, к которой подключен маршрутизатор.

В OSPF определены три типа сетей:

- Сети «точка-точка» (Point-to-Point networks);
- Широковещательные сети (Broadcast networks);
- Нешироковещательные сети (Non-Broadcast networks).

Сети «точка-точка» состоят из единственной пары маршрутизаторов. Примером таких сетей являются сети PPP, HDLS.

Широковещательные сети поддерживают подключение более двух маршрутизаторов и возможность адресации сообщения всем подключенными устройствам (широковещание). Примером широковещательной сети является Ethernet.

7. Понятие маршрутизации

Нешироковещательные сети поддерживают подключение более двух маршрутизаторов, но не имеют возможности адресации сообщения всем подключенным устройствам. В OSPF реализованы два режима нешироковещательных сетей. Первый режим, называемый нешироковещательный множественный доступ (non-broadcast multi-access, NBMA), имитирует работу OSPF в широковещательной сети. Второй режим, называемый «точка — много точек» (Point-to-MultiPoint), рассматривает нешироковещательную сеть как набор каналов «точка-точка». Примерами таких сетей являются Frame Relay, X.25.

По умолчанию OSPF-интерфейсы коммутаторов L3 D-Link поддерживают широковещательный тип сети (Broadcast). При желании администратор может изменить тип сети на интерфейсе на Point-to-Point с помощью специальной команды.

WAN-интерфейс маршрутизатора, подключенный к сети «точка-точка», имеет только одного соседа и должен установить только одно соседство. Ethernet-интерфейс коммутатора L3 подключен к сети с множественным доступом и может иметь множество соседей. Установление соседства со всеми устройствами в этом случае потребует много вычислительных ресурсов и породит большое количество трафика маршрутизации.

Разработчики OSPF создали концепцию, которая позволяет избежать необходимости установления соседства между всеми маршрутизаторами широковещательных сетей и сетей NBMA.

Каждая широковещательная сеть и сеть NBMA должна иметь **назначенный маршрутизатор** (*Designated Router, DR*) (рис. 7.30).

Он выполняет две важные функции.

1. Устанавливает соседство с каждым маршрутизатором в IP-сети и выполняет центральную роль в процессе синхронизации базы данных состояния канала. Он является центром сбора маршрутной информации.

2. Рассыпает Network-LSA для сети. В этих LSA перечисляется набор маршрутизаторов (включая Designated Router), подключенных к сети в настоящее время.

Designated Router является единой точкой отказа, поэтому для повышения отказоустойчивости в каждой широковещательной сети и сети NBMA выбирается **резервный назначенный маршрутизатор** (*Backup Designated Router, BDR*) (рис. 7.29). BDR, как и DR, устанавливает соседство со всеми маршрутизаторами сети и получает от них маршрутную информацию, однако в отличие от DR он не рассыпает Network-LSA для сети.

Один Designated Router и один Backup Designated Router выбираются для каждой IP-сети из подключенных к ней маршрутизаторов с помощью протокола OSPF Hello. Все маршрутизаторы сети устанавливают соседство только с DR и BDR (рис. 7.30). Между собой они соседство не устанавливают.

В случае выхода DR из строя BDR становится Designated Router, и в сети начинаются выборы нового BDR.

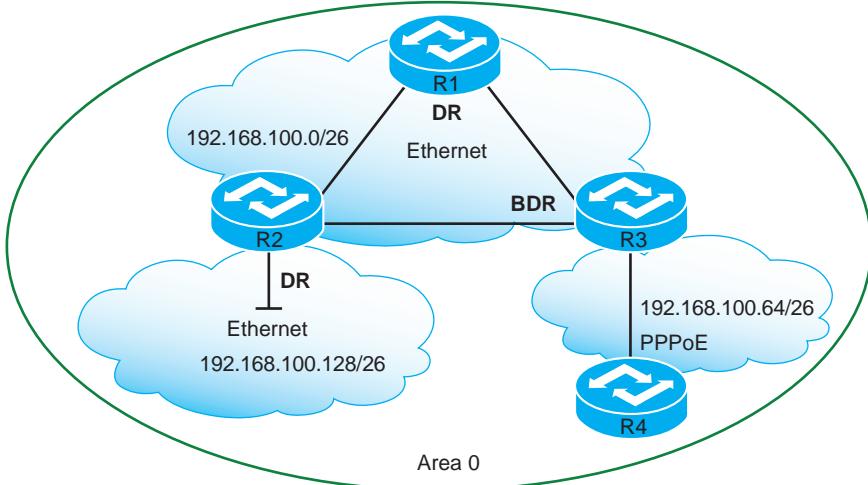


Рис. 7.29. DR и BDR

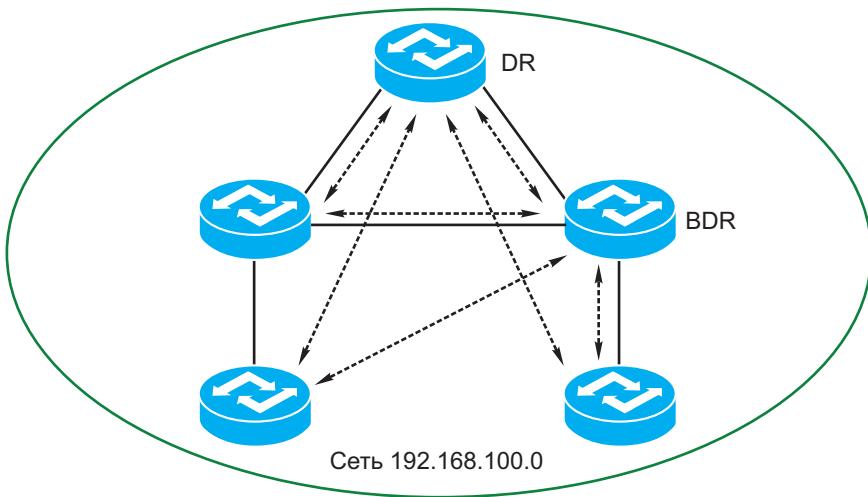


Рис. 7.30. Взаимосвязь DR и BDR с другими маршрутизаторами сети

В сетях «точка-точка» существуют только два узла, поэтому выборы DR и BDR в них не выполняются.

7.5.2. Типы пакетов протокола OSPF

Протокол OSPF (рис. 7.31) выполняется непосредственно поверх IP, используя номер протокола 89.

7. Понятие маршрутизации

```
Frame 204: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01), Dst: IPv4mcast_00:00:05 (01:00:5e:00:00:05)
Internet Protocol Version 4, Src: 192.168.10.3 (192.168.10.3), Dst: 224.0.0.5 (224.0.0.5)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 64
    Identification: 0x0002 (2)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: OSPF IGP (89)
    Header checksum: 0x0df3 [correct]
    Source: 192.168.10.3 (192.168.10.3)
    Destination: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
    OSPF Header
        OSPF Version: 2
        Message Type: Hello Packet (1)
        Packet Length: 44
        Source OSPF Router: 192.168.31.33 (192.168.31.33)
        Area ID: 0.0.0.0 (Backbone)
        Packet Checksum: 0x1cd5 [correct]
        Auth Type: Null
        Auth Data: (none)
OSPF Hello Packet
```

Рис. 7.31. Пакет протокола OSPF

Существуют пять типов пакетов протокола OSPF, которые позволяют устанавливать соседство и выполнять обновление маршрутной информации. Они перечислены в табл. 7.1.

Таблица 7.1. Типы пакетов OSPF

Тип пакета OSPF	Описание
Type 1 — Hello	Обнаруживает соседей и поддерживает соседские отношения
Type 2 — Database Description	Описывает содержимое базы данных состояния канала маршрутизатора
Type 3 — Link State Request (LSR)	Запрос на получение базы данных состояния канала
Type 4 — Link State Update (LSU)	Обновление базы данных состояния канала (передача LSA соседним маршрутизаторам)
Type 5 — Link State Acknowledgment (LSAck)	Подтверждение получения обновления

Протокол OSPF Hello использует пакеты Hello для обнаружения соседей и установления соседских отношений. Пакеты Database Description описывают топологию автономной системы или области. Пакеты Link State Request используются маршрутизатором для запроса обновленной информации о какой-то части базы данных состояния канала у другого маршрутизатора. Пакеты Link State Update содержат обновленную информацию о состоянии определенных интерфейсов в базе данных состояния каналов. Они отправляются маршрутизатором в ответ на сообщения Link State Request, а также

на регулярной основе путем многоадресной рассылки. Содержимое сообщений Link State Update используется для обновления информации в базах данных состояния канала маршрутизаторов, которые их получают. Сообщения Link State Acknowledgment служат для реализации надежного механизма обновления маршрутной информации. Они отправляются маршрутизаторами с целью подтверждения получения пакетов Link State Update.

Все типы пакетов OSPF используют одинаковый формат заголовка (рис. 7.32), который состоит из восьми полей:

- *Версия (Version)* — номер версии OSPF;
- *Тип (Type)* — тип пакета OSPF;
- *Длина пакета (Packet length)* — длина пакета в байтах, включая стандартный заголовок OSPF;
- *Идентификатор маршрутизатора (Router ID)* — идентификатор маршрутизатора, отправившего пакет;
- *Идентификатор области (Area ID)* — 32-битный номер, идентифицирующий область, которой этот пакет принадлежит. Все пакеты OSPF ассоциируются с единственной областью;
- *Контрольная сумма (Checksum)* — контрольная сумма всего содержимого пакета, начиная с заголовка, за исключением 64-битного поля «Аутентификация»;
- *Тип аутентификации (AuType)* — определяет метод аутентификации, используемый для пакета;

Биты: 0	7	15	31
Версия (Version)	Тип (Type)	Длина пакета (Packet length)	
Идентификатор маршрутизатора (Router ID)			
Идентификатор области (Area ID)			
Контрольная сумма (Checksum)		Тип аутентификации (AuType)	
Аутентификация (Authentication)			

Рис. 7.32. Формат заголовка пакета OSPF

- *Аутентификация (Authentication)* — 64-битное поле, используемое методом аутентификации.

Все пакеты OSPF (за исключение Hello) передаются только устройствам, с которыми установлены соседские отношения, т. е. в пределах одного шага.

Каждый пакет Link State Update, генерируемый маршрутизатором, содержит один или несколько LSA. Существует пять различных типов LSA. Их описание приведено в табл. 7.2.

7. Понятие маршрутизации

Таблица 7.2. Типы OSPF LSA

Тип пакета OSPF	Описание
Type 1 — Router-LSA	Генерируется каждым маршрутизатором для каждой области, которой он принадлежит. Он описывает состояние интерфейсов маршрутизатора, подключенных к этой области
Type 2 — Network-LSA	Генерируется назначенным маршрутизатором (DR). Он описывает набор маршрутизаторов, подключенных к определенной сети. Рассыпается только в области, содержащей эту сеть
Type 3 или 4 — Summary-LSA	Описывают маршруты между областями. Type 3 Summary-LSA генерируются ABR и описывают маршруты между ABR и внутренними маршрутизаторами локальной области. Они рассыпаются через магистраль другим ABR. Type 4 Summary-LSA описывают маршруты к ASBR
Type 5 — AS-external-LSA	Генерируются ASBR и описывают маршруты к пунктам назначения за пределами автономной системы. Маршрут по умолчанию для автономной системы также описывается AS-external-LSA

Каждое LSA начинается со стандартного 20-байтного заголовка. Он включает следующие поля (рис. 7.33):

- *LS age (возраст LS)*: это поле показывает количество секунд, прошедших с момента создания LSA. Когда LSA генерируется, значение этого поля равно 0. Оно увеличивается на каждом шаге в процессе рассылки (т. е. при передаче от одного соседнего маршрутизатора другому) и при хранении в базе данных состояния канала. Максимальное время жизни LSA равно MaxAge, значение которого установлено в 60 минут. LSA, достигшее максимального времени жизни, не используется при вычислении таблицы маршрутизации. Оно должно быть заново сгенерировано и отправлено соседям;
- *Options (опции)*: это поле показывает, какие дополнительные возможности ассоциированы с LSA;
- *LS type (тип LS)*: это поле определяет формат и тип LSA;
- *Link State ID (идентификатор состояния канала)*: это поле идентифицирует часть домена маршрутизации, который описывается LSA. В зависимости от типа LSA это поле может принимать различные значения;
- *Advertising Router (объявляющий маршрутизатор)*: это поле содержит Router ID маршрутизатора, сгенерировавшего LSA. Для Router-LSA это поле идентично полю *Link State ID*. Network-LSA генерируются DR. Summary-LSA порождаются ABR. AS-external-LSA генерируются ASBR;
- *LS sequence number (порядковый номер LS)*: этому полю присваивается 32-битное целое число. Оно используется для определения старых

и повторяющихся LSA. Порядковые номера линейно упорядочены. LSA с большим порядковым номером считается более актуальным;

- *LS checksum* (контрольная сумма LS): это поле является контрольной суммой всего содержимого LSA, за исключением поля *LS age*;
- *Length* (длина): это поле содержит длину LSA, включая заголовок.

Биты: 0	15 16	23 24	31
Возраст LS (LS age)	Опции (Options)	Тип LS (LS type)	
Идентификатор состояния канала (Link State ID)			
Объявляющий маршрутизатор (Advertising Router)			
Порядковый номер LS (LS sequence number)			
Контрольная сумма (LS checksum)	Длина (Length)		

Рис. 7.33. Формат заголовка LSA

Комбинация полей *LS type*, *Link State ID* и *Advertising Router* уникально идентифицирует LSA. Поля, следующие за заголовком, зависят от типа LSA, определенного в поле *LS type*.

7.5.3. Состояния соседства

Каждый маршрутизатор OSPF устанавливает отношения с соседними маршрутизаторами. Каждый диалог ограничен определенным интерфейсом маршрутизатора и идентифицируется или Router ID соседнего маршрутизатора, или его IP-адресом. Таким образом, если маршрутизатор подключен к множеству сетей, то он устанавливает множество соседских отношений, которые описываются уникальной структурой данных.

Процесс установления отношений между маршрутизаторами проходит через семь состояний.

1. **Состояние Down.** Это первоначальное состояние диалога между соседями. Соседи никакой информацией не обмениваются и ждут перехода в следующее состояние Init.

2. **Состояние Init.** Для установления соседских отношений маршрутизаторы через регулярные интервалы отправляют пакеты Hello. Как только интерфейс получил первый пакет Hello, маршрутизатор переходит в состояние Init. Получая пакеты Hello от других маршрутизаторов, он узнает, что где-то есть соседи, и ожидает, что отношения перейдут в следующую фазу. На этом этапе еще не установлены двухсторонние отношения между соседями.

3. **Состояние 2-Way.** В этом состоянии устанавливаются двухсторонние отношения между маршрутизаторами, находящимися в одной сети. Это выполняется с помощью протокола OSPF Hello. Как только маршрутизатор получает пакет Hello от соседа и видит себя в списке известных ему

маршрутизаторов, двухсторонние отношения считаются установленными. Также в этом состоянии из набора соседей происходят выборы DR и BDR.

4. Состояние ExStart. Это первый шаг к установлению соседства между двумя соседними маршрутизаторами. Цель этого состояния — определить, какой маршрутизатор будет играть роль мастера (master), а какой — ведомого (slave), и договориться о первоначальном порядковом номере пакета Database Description. Состояние диалога между соседями в этом и последующих шагах называется соседством (adjacency).

5. Состояние Exchange. В этом состоянии маршрутизатор описывает свою полную базу данных состояния канала, отправляя последовательность пакетов Database Description соседу. Каждый пакет Database Description имеет порядковый номер и требует подтверждения. В один момент времени можно отправлять только один пакет Database Description. Получив от соседей информацию и проанализировав ее, каждый маршрутизатор знает, какая часть базы данных состояния канала в настоящий момент не актуальна или отсутствует. Поэтому в этом состоянии также может быть отправлен пакет Link State Request с целью запроса у соседа последних LSA. На данном и последующих этапах маршрутизаторы, установившие соседство, могут обмениваться всеми типами пакетов OSPF.

6. Состояние Loading. В этом состоянии каждый маршрутизатор может запросить у соседа актуальные LSA, которые были обнаружены (но не получены) в состоянии Exchange, отправив пакет Link State Request.

7. Состояние Full. В этом состоянии соседние маршрутизаторы установили полное соседство (fully adjacent). Эти соседства теперь будут появляться в Router-LSA и Network-LSA.

7.5.4. Установление соседства

Чтобы обмениваться маршрутной информацией в сети OSPF, маршрутизаторы должны установить соседство. За установку и поддержку соседских отношений отвечает протокол OSPF Hello. Он гарантирует, что взаимодействие между соседями будет двухстороннее. В широковещательных сетях и сетях NBMA протокол используется для выбора Designated Router и Backup Designated Router.

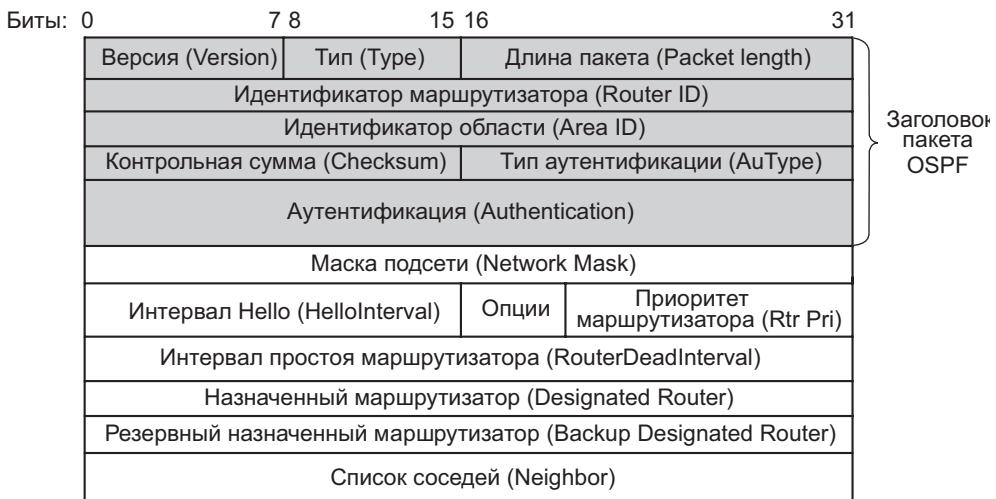
Как только маршрутизаторы становятся активны, они начинают отправлять на регулярной основе через каждый работающий интерфейс пакеты Hello. Формат пакета показан на рис. 7.34.

Пакет Hello содержит следующие поля:

- *Маска подсети (Network mask).* Мaska подсети, к которой подключен интерфейс;

- *Интервал Hello (HelloInterval).* Интервал времени в секундах между отправками пакетов Hello с интерфейса маршрутизатора. По умолчанию составляет 10 секунд;

- *Опции (Options).* Дополнительные возможности, поддерживаемые маршрутизатором;

**Рис. 7.34.** Формат пакета Hello

- *Приоритет маршрутизатора (Router Priority)*. Приоритет маршрутизатора, используемый в процессе выборов DR и BDR. Если его значение равно 0, маршрутизатор не может быть выбран DR или BDR;
- *Интервал простоя маршрутизатора (RouterDeadInterval)*. Интервал времени в секундах, который маршрутизатор ожидает, прежде чем решит, что соседний маршрутизатор неактивен (т. е. если от соседа в этот период не приходят пакеты Hello). По умолчанию составляет 40 секунд.

Оба интервала HelloInterval и RouterDeadInterval могут настраиваться администратором сети, и они должны совпадать на всех маршрутизаторах, подключенных к одной сети, но при этом HelloInterval должен быть меньше RouterDeadInterval. Если два маршрутизатора имеют разные интервалы HelloInterval и RouterDeadInterval, они не смогут установить соседство и обмениваться маршрутной информацией;

- *Назначенный маршрутизатор (Designated Router)*. Идентификатор DR для этой сети с точки зрения маршрутизатора, отправляющего пакет. Здесь DR идентифицируется с помощью IP-адреса интерфейса, подключенного к сети. Если в сети нет DR, значение этого поля 0.0.0.0;

• *Резервный назначенный маршрутизатор (Backup Designated Router)*. Идентификатор BDR для этой сети с точки зрения маршрутизатора, отправляющего пакет. Здесь BDR идентифицируется с помощью IP-адреса интерфейса, подключенного к сети. Если в сети нет BDR, значение этого поля 0.0.0.0;

- *Список соседей (Neighbor)*. Идентификаторы маршрутизаторов (Router ID), от которых в течение последнего интервала RouterDeadInterval были получены пакеты Hello.

Пакеты Hello отправляются на групповой адрес 224.0.0.5, включающий все маршрутизаторы OSPF.

7. Понятие маршрутизации

На рис. 7.35 показан пример сети OSPF. Рассмотрим процесс установления соседства между коммутаторами L3, подключенными к сети 192.168.10.0/24. После активизации коммутаторы SW1, SW2 и SW3 начинают рассыпать пакеты Hello через все активные интерфейсы, объявляя свои Router ID. Идентификаторы Router ID вручную настраивает администратор сети.

Предположим, что все коммутаторы настроены правильно, поэтому в сети

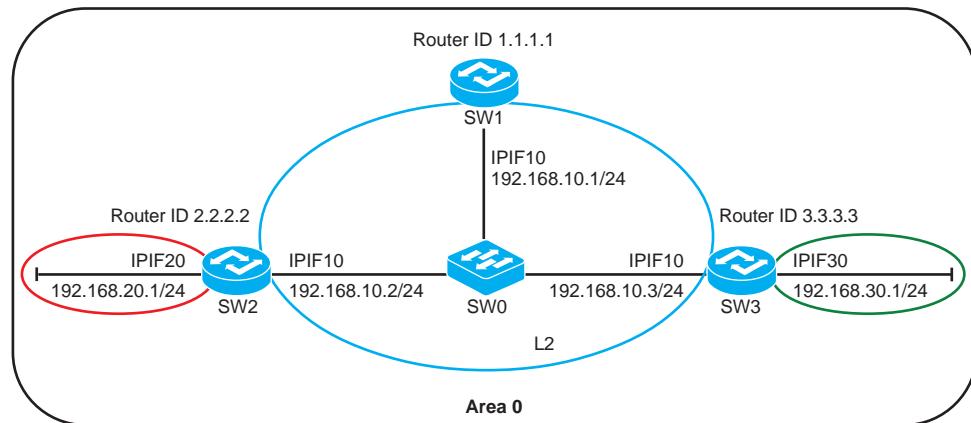


Рис. 7.35. Пример сети OSPF

192.168.10.0/24 коммутатор SW1 получит пакеты Hello от SW2 и SW3, SW2 получит пакеты Hello от SW1 (рис. 7.36) и SW3, SW3 получит пакеты Hello от SW1 и SW2. Как только каждый из коммутаторов получит первый пакет Hello от соседа, он проверит, совпадают ли Area ID, маска подсети, параметры таймеров с его собственными.

```
Frame 4/: 8 bytes on wire (624 bits), 8 bytes captured (624 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: IPv4mcast_00:00:05 (01:00:5e:00:00:05)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
  OSPF Header
    OSPF Version: 2
    Message Type: Hello Packet (1)
    Packet Length: 44
    Source OSPF Router: 1.1.1.1 (1.1.1.1)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0xfa9c [correct]
    Auth Type: Null
    Auth Data (none)
  OSPF Hello Packet
    Network Mask: 255.255.255.0
    Hello Interval: 10 seconds
    Options: 0x02 (E)
    Router Priority: 1
    Router Dead Interval: 40 seconds
    Designated Router: 0.0.0.0
    Backup Designated Router: 0.0.0.0
```

Рис. 7.36. Первый пакет Hello, отправленный коммутатором SW1 (Router ID 1.1.1.1)

Если параметры совпадают, коммутатор перейдет в состояние Init, и в следующих отправляемых пакетах Hello (рис. 7.37) будет включать в поле *Neighbor* идентификаторы обнаруженных соседних маршрутизаторов.

```
Frame 61: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: IPv4mcast_00:00:05 (01:00:5e:00:00:05)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
OSPF Header
    OSPF Version: 2
    Message Type: Hello Packet (1)
    Packet Length: 52
    Source OSPF Router: 1.1.1.1 (1.1.1.1)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0xf08a [correct]
    Auth Type: Null
    Auth Data (none)
OSPF Hello Packet
    Network Mask: 255.255.255.0
    Hello Interval: 10 seconds
    Options: 0x02 (E)
    Router Priority: 1
    Router Dead Interval: 40 seconds
    Designated Router: 0.0.0.0
    Backup Designated Router: 0.0.0.0
    Active Neighbor: 3.3.3.3
    Active Neighbor: 2.2.2.2
```

Рис. 7.37. Пакет Hello со списком обнаруженных соседей

Когда коммутатор получит пакет Hello и увидит свой идентификатор в поле *Neighbor*, значит, он установил двухсторонние отношения с соседом, отправившим этот пакет.

Предположим, что коммутаторы SW1, SW2 и SW3 установили между собой двухсторонние отношения (состояние 2-Way). Следующим их шагом является определение устройств, с которыми будут установлены соседские отношения.

С какими устройствами будут установлены соседские отношения, зависит от типа сети, к которой подключен определенный интерфейс. Если сеть «точка-точка», то маршрутизирующее устройство установит соседство со своим единственным партнером по связи. Если сеть широковещательная, то потребуются выборы DR и BDR.

В нашем примере коммутаторы подключены к сети Ethernet, которая является широковещательной. Поэтому прежде чем будет установлено соседство, необходимо в каждой сети выбрать DR и BDR.

Выборы DR и BDR

Выборы DR и BDR выполняются только в широковещательной сети или сети NBMA. В первоначальный момент времени поля *Designated Router* и *Backup Designated Router* в пакетах Hello (рис. 7.38) имеют значение 0.0.0.0. Это означает, что DR и BDR не выбраны. Выборы DR и BDR выполняются с помощью протокола OSPF Hello после того, как маршрутизаторы

7. Понятие маршрутизации

установили двухсторонние отношения со своими соседями. Пакеты Hello маршрутизаторов содержат информацию об их Router ID и приоритете (Router Priority), который используется в процессе выборов.

Каждый маршрутизатор изучает список соседей, с которыми он установил двухсторонние отношения в данной сети. Себя он также включает в этот список. Для начала из списка удаляются все маршрутизаторы, которые не могут быть выбраны на роли DR и BDR. У таких маршрутизаторов значение приоритета равно 0.

Среди оставшихся маршрутизаторов выбирается маршрутизатор с наивысшим значением приоритета. Он выигрывает выборы и становится DR. Маршрутизатор со вторым по значению наивысшим значением приоритета выбирается на роль BDR.

По умолчанию все маршрутизаторы OSPF имеют значение приоритета, равное 1. Администратор сети может вручную назначать приоритеты маршрутизаторам и влиять на процесс выборов. Приоритет, равный 0, исключает маршрутизатор из участия в выборах.

Если несколько маршрутизаторов имеют одинаковый приоритет, то выборы выигрывает маршрутизатор с наибольшим значением Router ID.

Необходимо понимать, что DR и BDR выбираются в каждой широковещательной сети. Область OSPF может состоять из нескольких широковещательных сетей или подсетей, таким образом, в ней будет существовать несколько DR и BDR. Даже если к сети подключен только один маршрутизатор, DR все равно выбирается, поскольку потенциально к ней может быть подключен еще один или несколько маршрутизаторов.

Когда DR и BDR выбраны, они сохраняют свои роли до тех пор, пока один из них или оба они не выйдут из строя или отключатся. Даже если в сеть добавляется маршрутизатор с самым большим значением приоритета, процесс выборов не запускается. Новый маршрутизатор получает информацию об идентификаторах DR и BDR из пакетов Hello, рассылаемых другими маршрутизаторами сети. В них DR и BDR идентифицируются с помощью IP-адреса интерфейса, подключенного к сети.

Вернемся к рассмотрению примера сети OSPF, показанной на рис. 7.35. Несмотря на то что к сетям 192.168.20.0/24 и 192.168.30.0/24 подключен только один коммутатор L3, в них все равно выполняются выборы DR. В сети 192.168.20.0/24 роль DR будет играть коммутатор SW2, в сети 192.168.30.0/24 роль DR — коммутатор SW3.

К сети 192.168.10.0/24 подключены три коммутатора L3. Так как приоритет у коммутаторов SW1, SW2 и SW3 равен 1, выборы DR и BDR будут выполняться на основе сравнения Router ID. Наивысший Router ID, равный 3.3.3.3, имеет SW3, поэтому он выбирается на роль DR. Второй по значению Router ID, равный 2.2.2.2, у SW2, поэтому он становится BDR. Следует отметить, что SW3 играет роль DR в сетях 192.168.10.0/24 и 192.168.30.0/24. Коммутатор SW2 выполняет две функции: DR в сети 192.168.20.0/24 и BDR в сети 192.168.10.0/24.

Технологии TCP/IP в современных компьютерных сетях

```
Frame 118: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
Ethernet II, Src: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01), Dst: IPv4mcast_00:00:05 (01:00:5e:00:00:05)
Internet Protocol Version 4, Src: 192.168.10.3 (192.168.10.3), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
  OSPF Header
    OSPF Hello Packet
      Network Mask: 255.255.255.0
      Hello Interval: 10 seconds
      Options: 0x02 (E)
      Router Priority: 1
      Router Dead Interval: 40 seconds
      Designated Router: 192.168.10.3
      Backup Designated Router: 192.168.10.2
      Active Neighbor: 1.1.1.1
      Active Neighbor: 2.2.2.2
```

Рис. 7.38. Пакет Hello с идентификаторами DR и BDR

Когда двухсторонние отношения установлены и выборы DR/BDR завершены, маршрутизаторы готовы к обмену базами данных состояния канала с соседними маршрутизаторами.

Синхронизация баз данных состояния канала

В протоколе маршрутизации с учетом состояния канала очень важно, чтобы базы данных состояния канала всех маршрутизаторов были синхронизированы. Протокол OSPF требует, чтобы были синхронизированы только базы данных маршрутизаторов, установивших соседство. Процесс синхронизации начинается сразу, как только маршрутизаторы попытались установить соседство. Каждый маршрутизатор описывает свою базу данных, отправляя соседям последовательность пакетов Database Description. Пакеты Database Description отправляются на индивидуальный адрес конкретного соседа. Каждый пакет Database Description содержит набор LSA из базы данных маршрутизатора. Когда сосед получает пакет Database Description и видит в нем LSA, который более актуален, чем хранимый в его базе данных, он делает себе отметку, что требуется запросить этот обновленный LSA.

Отправка и получение пакетов Database Description называется *процессом обмена базами данных* (Database Exchange Process). На первом шаге этого процесса два соседних маршрутизатора формируют отношения мастер/ведомый (master/slave), для того чтобы определить, кто первым начнет обмен. Каждый пакет Database Description содержит порядковый номер. Мастер первым начинает отправлять пакеты Database Description. За один раз можно отправить только один пакет. Каждый пакет Database Description, отправляемый мастером (опрос), должен подтверждаться ведомым маршрутизатором путем повторения в ответном пакете Database Description полученного порядкового номера. Пакеты Database Description, отправляемые мастером и ведомым маршрутизатором, содержат краткое описание их полных баз данных состояния канала. Повторно передавать пакеты Database Description разрешено только мастеру. Он может это делать только через фиксированные интервалы времени.

Отношения мастер/ведомый устанавливаются в состоянии ExStart. Также в этом состоянии маршрутизаторы договариваются о первоначальном

7. Понятие маршрутизации

порядковом номере пакета Database Description. Когда маршрутизатор отправляет пакет Database Description в первый раз (рис. 7.39), он присваивает уникальное значение первоначальному порядковому номеру пакета, объявляется мастером, устанавливая бит Master (MS) в 1, а также устанавливает биты Initialize (I) и More (M). Первый отправляемый пакет Database Description должен быть пустым. Этот пакет повторно передается через определенные интервалы времени до тех пор, пока маршрутизатор не перейдет в следующее состояние.

```
Frame 116: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01), Dst: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80)
Internet Protocol Version 4, Src: 192.168.10.3 (192.168.10.3), Dst: 192.168.10.1 (192.168.10.1)
Open Shortest Path First
  OSPF Header
  OSPF DB Description
    Interface MTU: 1500
    Options: 0x02 (E)
    DB Description: 0x07 (I, M, MS)
      .... 0... = R: OOBResync bit is NOT set
      .... .1.. = I: Init bit is SET
      .... ..1. = M: More bit is SET
      .... .1 = MS: Master/Slave bit is SET
DD Sequence: 173693708
```

Рис. 7.39. Первоначальный пакет Database Description

Когда маршрутизатор получает от соседа пустой пакет Database Description, он проверяет, установлены ли биты I, M и MS и сравнивает свой Router ID с идентификатором соседа, отправившего пакет. Мастером становится тот маршрутизатор, чей Router ID выше. Маршрутизатор, ставший ведомым, устанавливает в своих последующих пакетах Database Description бит MS в 0, а значение порядкового номера пакета меняет на значение, установленное мастером.

Вернемся к примеру сети OSPF (рис. 7.40). Напомним, что в результате выборов в сети 192.168.10.0/24 роль DR выполняет SW3, роль BDR — SW2.

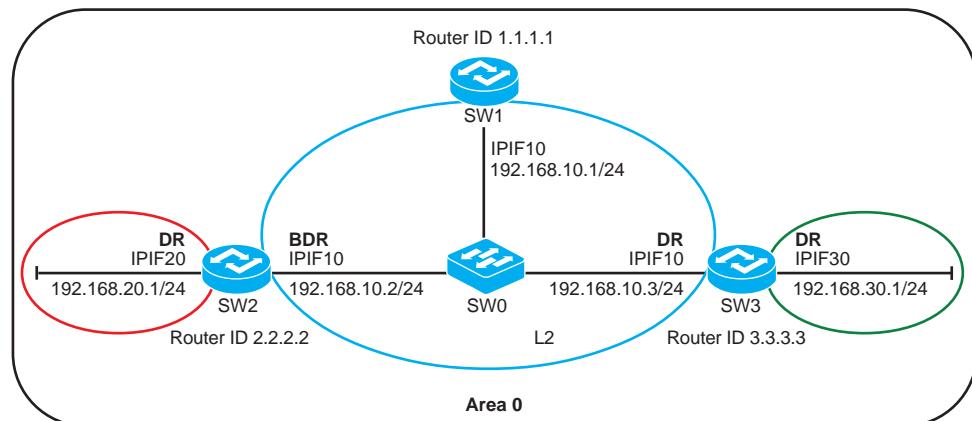


Рис. 7.40. Продолжение примера сети OSPF (см. рис. 7.35)

В широковещательной сети все маршрутизаторы устанавливают соседские отношения только с DR и BDR. В данном примере SW1 начал устанавливать соседские отношения с SW2 и SW3; SW2 начал устанавливать соседские отношения с SW1 и SW3; SW3 начал устанавливать соседские отношения с SW1 и SW2. Обмен пакетами Database Description выполняется непосредственно между двумя маршрутизаторами, и они между собой решают, кто будет мастером, а кто — ведомым. Исходя из установленного соседства, SW1 отправит пустые пакеты Database Description коммутаторам SW2 и SW3 на их индивидуальные адреса; SW2 отправит пустые пакеты Database Description коммутаторам SW1 и SW3; SW3 отправит пустые пакеты Database Description коммутаторам SW1 и SW2. В паре коммутаторов SW1-SW2 мастером станет SW2, так как его Router ID выше; в паре SW1-SW3 мастером станет SW3; в паре SW2-SW3 мастером также станет SW3.

После того как маршрутизаторы определили, кто будет мастером, а кто — ведомым, они переходят в состояние Exchange. В этом состоянии маршрутизаторы кратко описывают свои полные базы данных состояния канала, отправляя на индивидуальный адрес соседа пакеты Database Description. Краткое описание включает тип LSA, идентификатор маршрутизатора, отправившего LSA, порядковый номер LSA.

Обмен начинает мастер, отправляя ведомому маршрутизатору пакет Database Description, содержащий описание базы данных и порядковый номер пакета. Ведомый маршрутизатор отвечает пакетом Database Description, в который помещает краткое описание своей базы данных и порядковый номер из пакета мастера. Пакетов для передачи может быть несколько. Порядковый номер каждого нового пакета Database Description, передаваемого мастером, увеличивается на 1 до тех пор, пока бит M, содержащийся в пакете, не станет равным 0. Бит M показывает, имеются ли пакеты для передачи или нет. Если бит M равен 1, то этот пакет не последний и обмен не завершен. Когда маршрутизатор получит и отправит пакет Database Description с битом M, равным 0, это будет означать, что процесс обмена базами данных завершен.

В состоянии Exchange маршрутизаторы обмениваются друг с другом кратким описанием баз данных состояния канала. Получая информацию, каждый маршрутизатор сравнивает ее с информацией, хранимой в его базе данных. Если полученная информация отсутствует в его базе данных или она более актуальная, маршрутизатор запрашивает у соответствующего соседа требуемые LSA, отправляя на его индивидуальный адрес пакет Link State Request.

Завершается обмен маршрутной информацией в состоянии Loading. Когда маршрутизатор получает от соседа запрос Link State Request, содержащий список требуемых LSA, в ответ он отправляет пакет Link State Update, содержащий их подробное описание. Получение пакета Link State Update требует подтверждения путем отправки пакета Link State Ack.

Когда на все отправленные запросы Link State Request получены ответы, базы данных маршрутизаторов будут полностью синхронизированы, и они перейдут в состояние полного соседства (Full).

7. Понятие маршрутизации

```
Frame 231: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: ac:f1:df:b5:fc:00 (ac:f1:df:b5:fc:00)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.2 (192.168.10.2)
Open Shortest Path First
  OSPF Header
  OSPF DB Description
    Interface MTU: 1500
    Options: 0x02 (E)
    DB Description: 0x00
    DD Sequence: 173693753
  LSA Header
    LS Age: 8 seconds
    Do Not Age: False
    Options: 0x02 (E)
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 1.1.1.1
    Advertising Router: 1.1.1.1 (1.1.1.1)
    LS Sequence Number: 0x80000003
    LS Checksum: 0x411d
    Length: 36
  LSA Header
    LS Age: 4 seconds
    Do Not Age: False
    Options: 0x02 (E)
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 2.2.2.2
    Advertising Router: 2.2.2.2 (2.2.2.2)
    LS Sequence Number: 0x80000005
    LS Checksum: 0xcf5
```

Рис. 7.41. Пакет Database Description, содержащий набор LSA

```
Frame 133: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.3 (192.168.10.3)
Open Shortest Path First
  OSPF Header
  Link State Request
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 3.3.3.3
    Advertising Router: 3.3.3.3 (3.3.3.3)
```

Рис. 7.42. Пакет Link State Request

```
Frame 140: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
Ethernet II, Src: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01), Dst: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80)
Internet Protocol Version 4, Src: 192.168.10.3 (192.168.10.3), Dst: 192.168.10.1 (192.168.10.1)
Open Shortest Path First
  OSPF Header
  LS Update Packet
    Number of LSAs: 1
    LS Type: Router-LSA
      LS Age: 1 seconds
      Do Not Age: False
      Options: 0x02 (E)
      Link-State Advertisement Type: Router-LSA (1)
      Link State ID: 3.3.3.3
      Advertising Router: 3.3.3.3 (3.3.3.3)
      LS Sequence Number: 0x80000004
      LS Checksum: 0x67c3
      Length: 48
    Flags: 0x00
    Number of Links: 2
    Type: Stub      ID: 192.168.30.0      Data: 255.255.255.0      Metric: 1
    Type: Stub      ID: 192.168.10.0      Data: 255.255.255.0      Metric: 1
```

Рис. 7.43. Пакет Link State Update

Технологии TCP/IP в современных компьютерных сетях

```
Frame 142: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: ac:f1:df:b6:03:80 (ac:f1:df:b6:03:80), Dst: 84:c9:b2:1f:9c:01 (84:c9:b2:1f:9c:01)
Internet Protocol Version 4, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.3 (192.168.10.3)
Open Shortest Path First
  OSPF Header
    OSPF Version: 2
    Message Type: LS Acknowledge (5)
    Packet Length: 44
    Source OSPF Router: 1.1.1.1 (1.1.1.1)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0x023a [correct]
    Auth Type: Null
    Auth Data (none)
  LSA Header
    LS Age: 5 seconds
    Do Not Age: False
  Options: 0x02 (E)
    Link-State Advertisement Type: Router-LSA (1)
    Link State ID: 1.1.1.1
    Advertising Router: 1.1.1.1 (1.1.1.1)
    LS Sequence Number: 0x80000002
    LS Checksum: 0x7362
    Length: 36
```

Рис. 7.44. Пакет Link State Ack

Command: show ospf neighbor

IP Address of Neighbor	Router ID of Neighbor	Neighbor Priority	Neighbor State
192.168.10.2	2.2.2.2	1	Full
192.168.10.3	3.3.3.3	1	Full

Total Entries : 2

Рис. 7.45. Список соседей SW1

Command: show ospf lsdb

Area ID	LSDB Type	Advertising Router ID	Link State ID	Cost	Sequence Number
0.0.0.0	RTRLink	1.1.1.1	1.1.1.1/0	*	0x80000003
0.0.0.0	RTRLink	2.2.2.2	2.2.2.2/0	*	0x80000003
0.0.0.0	RTRLink	3.3.3.3	3.3.3.3/0	*	0x80000003
0.0.0.0	NETLink	3.3.3.3	3.3.3.3/24	*	0x80000002

Total Entries: 4

Рис. 7.46. База данных состояния канала на SW1, SW2, SW3

7. Понятие маршрутизации

Вернемся к рассматриваемому примеру. После обмена пакетами Database Description и удовлетворения запросов Link State Request коммутаторы SW1, SW2 и SW3 установили полное соседство, и они должны иметь одинаковые базы данных состояния канала (рис. 7.45, 7.46).

Следующим шагом после установления соседства является создание таблиц маршрутизации.

7.5.5. Вычисление маршрутов

После того как маршрутизаторы синхронизировали свои базы данных, они готовы к созданию таблиц маршрутизации.

В сети OSPF существуют четыре типа маршрутов:

- Intra-area (маршруты внутри области): маршруты, ведущие к пунктам назначения внутри областей, к которым подключен маршрутизатор;
- Inter-area (маршруты между областями): маршруты к пунктам назначения в других областях. Обнаруживаются путем исследования получаемых Summary-LSA;
- Type 1 external и Type 2 external: маршруты к пунктам назначения, лежащим за пределами автономной системы. Обнаруживаются путем исследования получаемых AS-external-LSA.

Используя базы данных состояния канала подключенных областей, маршрутизатор выполняет алгоритм для создания таблицы маршрутизации:

1. Существующая таблица маршрутизации признается недействительной.

Таблица маршрутизации строится заново. Старая таблица сохраняется, чтобы можно было обнаружить изменения в записях.

2. Маршруты внутри области вычисляются маршрутизатором путем построения дерева кратчайших маршрутов для каждой области, к которой он подключен. Создание дерева кратчайших маршрутов внутри области состоит из двух фаз. На первой фазе построения дерева рассматриваются только каналы между маршрутизаторами и транзитными сетями. Используя алгоритм Dijkstra, дерево формируется из этого поднабора данных, содержащихся в базе данных состояния канала. На второй фазе к дереву добавляются листья путем рассмотрения каналов к тупиковым сетям. В процессе построения дерева для области вычисляется TransitCapability, используемая на шаге 4.

3. Маршруты между областями вычисляются на основе исследования Summary-LSA. Если маршрутизатор является ABR, исследуются только Summary-LSA магистральной области.

4. Маршрутизаторы ABR, подключенные к одной или нескольким транзитным областям (не магистральным, с TransitCapability, равной TRUE), исследуют Summary-LSA этих областей, чтобы определить лучшие, чем были найдены на шагах 2–3, маршруты через транзитные области.

5. Вычисляются маршруты к пунктам назначения за пределами автономной системы. Для этого анализируются AS-external-LSA. Местоположение граничных маршрутизаторов AS (ASBR) было определено на шагах 2–4.

Вычисление кратчайшего пути внутри области

Маршрутизация внутри области определяется ее собственной топологией. Как только соседи синхронизировали базы данных состояния канала, каждый маршрутизатор с помощью алгоритма Dijkstra вычисляет дерево кратчайших маршрутов (shortest path tree) через область. Себя маршрутизатор использует в качестве корня. Следовательно, дерево кратчайших маршрутов у разных маршрутизаторов будет различаться. Дерево определяет кратчайший путь к любой сети или узлу в пределах области. Если маршрутизатор подключен к разным областям, он вычисляет дерево кратчайших маршрутов для каждой области.

Когда алгоритм Dijkstra пытается определить наилучший маршрут к пункту назначения, он вычисляет и сравнивает стоимость различных путей к нему от корня. Стоимость пути (Cost) вычисляется как сумма стоимостей каждого из интерфейсов OSPF, находящихся на маршруте между корнем и пунктом назначения. С каждым интерфейсом OSPF связана стоимость (также называемая метрикой (metric)), которая показывает затраты на отправку данных через него. По умолчанию стоимость для интерфейсов OSPF на L3-коммутаторах D-Link равна 1. Администратор сети может изменить ее с помощью соответствующих настроек.

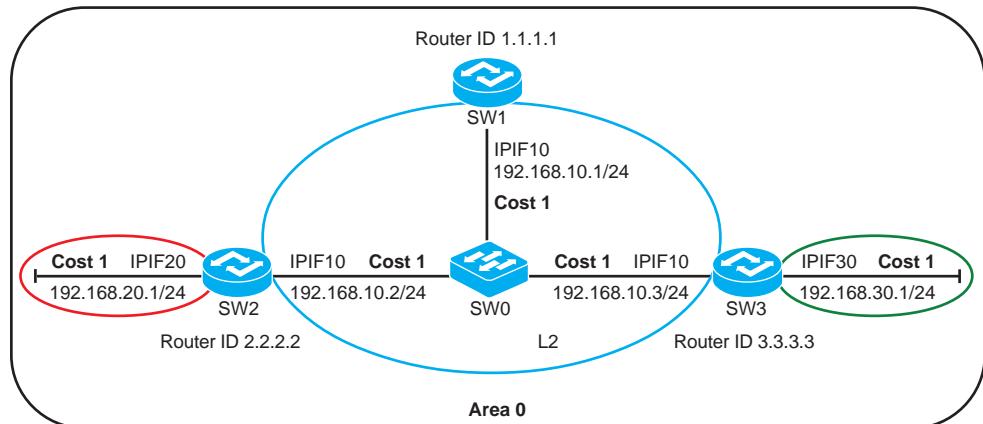


Рис. 7.47. Продолжение примера для сети OSPF

Если маршрутов к пункту назначения несколько, то предпочтительным является маршрут с наименьшей стоимостью, который будет кратчайшим. Он заносится в таблицу маршрутизации OSPF. В результате вычислений может оказаться, что существуют несколько кратчайших маршрутов. Для балансировки нагрузки L3-коммутаторы D-Link могут помещать несколько маршрутов с одинаковой стоимостью в таблицу маршрутизации. Количество сохраняемых маршрутов зависит от модели устройства.

7. Понятие маршрутизации

Давайте вернемся к рассматриваемому примеру сети OSPF (рис. 7.47). Для вычисления маршрутов внутри области Area 0 каждый коммутатор L3 анализирует базу данных состояния канала и определяет все сети внутри области.

Command: `show iproute`

Routing Table

IP Address/Netmask	Gateway	Interface	Cost	Protocol
192.168.10.0/24	0.0.0.0	IPIF10	1	Local
192.168.20.0/24	192.168.10.2	IPIF10	2	OSPF
192.168.30.0/24	192.168.10.3	IPIF10	2	OSPF

Total Entries: 3

Рис. 7.48. Таблица маршрутизации SW1

В нашем примере существуют три сети: 192.168.10.0/24, 192.168.20.0/24 и 192.168.30.0/24. Каждый из коммутаторов для каждого из возможных маршрутов вычисляет суммарные стоимости, складывая стоимости интерфейсов. Например, стоимость пути от SW1 до сети 192.168.10.0/24 равна 1 (cost IPIF10); стоимость пути до сети 192.168.20.0/24 через SW2 равна 2 (cost IPIF10 + cost IPIF20); стоимость пути до сети 192.168.30.0/24 через SW3 равна 2 (cost IPIF10 + cost IPIF30). К каждой из сетей от SW1 существует только один маршрут, поэтому выбирать наилучший не требуется и все три найденных маршрута будут помещены в таблицу маршрутизации (рис. 7.48).

7.5.6. Обновление маршрутной информации внутри области

Все маршрутизаторы должны отслеживать маршрутную информацию и оперативно вносить изменения в таблицу маршрутизации. Когда происходит какое-то событие, маршрутизатор генерирует и рассыпает своим соседям LSA, сообщая о произошедших изменениях. Напомним, что существуют пять типов LSA, которые может генерировать маршрутизатор внутри любой области OSPF. Router-LSA и Network-LSA используются для рассылки внутри области. Router-LSA может рассыпать любой маршрутизатор. Network-LSA рассыпает Designated Router.

Маршрутизатор OSPF генерирует Router-LSA или Network-LSA в нескольких случаях:

- изменилось состояние интерфейса;
- изменился Designated Router;
- изменилось состояние соседнего маршрутизатора;
- истекло время жизни LSA. Каждая запись LSA имеет время жизни.

Если оно достигло максимального значения, маршрутизатор генерирует

новое LSA, даже если изменений не произошло. Это позволяет периодически обновлять информацию.

Когда маршрутизатор генерирует LSA, он помещает его в пакет Link State Update и отправляет своим соседям. В сетях «точка-точка» нет DR и BDR, поэтому маршрутизатор генерирует Router-LSA, помещает его в пакет Link State Update и отправляет на групповой адрес 224.0.0.5 (все маршрутизаторы OSPF). При получении пакета Link State Update сосед должен отправить подтверждение Link State Ack.

В широковещательных сетях имеются DR и BDR, которые поддерживают соседство со всеми маршрутизаторами данной сети/подсети. Если изменения обнаружил DR или BDR, он генерирует Network-LSA, помещает его в пакет Link State Update и отправляет на групповой адрес 224.0.0.5. Каждый маршрутизатор сети должен отправить подтверждение при получении пакета Link State Update.

Если событие, вызвавшее изменение состояния, произошло на другом маршрутизаторе широковещательной сети (рис. 7.49), он генерирует Router-LSA, помещает его в пакет Link State Update и отправляет на групповой адрес 224.0.0.6 (все маршрутизаторы DR/BDR). Когда DR получает это сообщение,

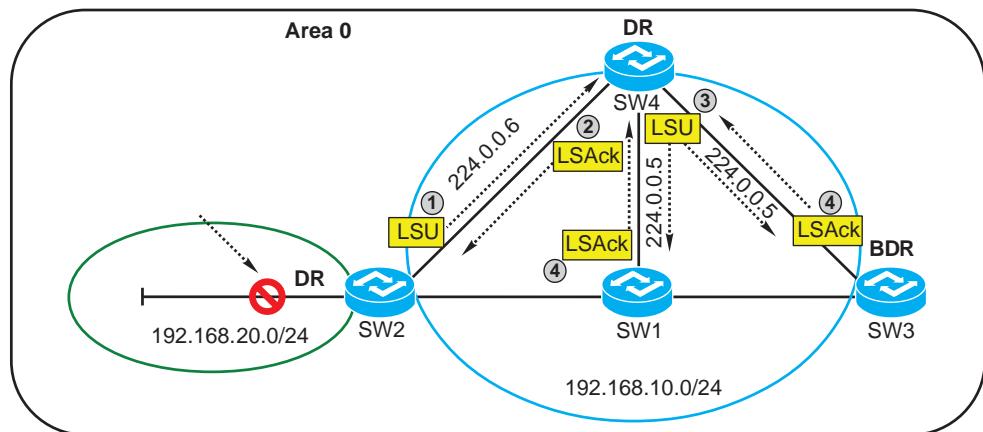


Рис. 7.49. Распространение обновленной маршрутной информации внутри области

он посылает подтверждение. Далее DR генерирует Network-LSA, помещает его в пакет Link State Update и отправляет на групповой адрес 224.0.0.5, чтобы его получили все остальные маршрутизаторы сети. Маршрутизаторы должны подтвердить получение.

Если маршрутизатор подключен к разным сетям, принадлежащим одной области, он отправляет пакет Link State Update, содержащий LSA, во все подключенные сети.

После получения пакета Link State Update, содержащего LSA с обновленной информацией, маршрутизатор обновляет свою базу данных состояния канала. Используя новую информацию, с помощью алгоритма Dijkstra вычисляются новые кратчайшие маршруты, которые заносятся в таблицу маршрутизации.

7.6. Протокол OSPF версии 3

В 1999 году была опубликована третья версия протокола OSPF, которая поддерживает IPv6. Первоначально OSPF version 3 (OSPFv3) была описана в RFC 2740, позже обновлена в RFC 5340. Фундаментальные механизмы OSPF (рассылка, выборы DR, поддержка областей, вычисление кратчайших маршрутов и т. д.) в новой версии протокола остались прежними. При этом стоит отметить, что OSPFv2 и OSPFv3 обратно не совместимы. Поэтому если необходимо использовать OSPF для маршрутизации пакетов IPv4 и IPv6, то на маршрутизаторе необходимо включить обе версии протокола.

Рассмотрим основные отличия протокола OSPFv3 от OSPFv2.

1. **Работа протокола на канале.** Протокол IPv6 использует термин «канал» (link) для обозначения «канала связи или среды, через которую узлы могут взаимодействовать на канальном уровне». Интерфейсы подключаются к каналам. Один канал может принадлежать нескольким подсетям IPv6. При этом два узла, непосредственно подключенные к одному каналу, могут обмениваться информацией, даже если принадлежат разным подсетям (имеют разный префикс IPv6). OSPFv3 заменяет термины «сеть» и «подсеть», используемые в OSPFv2 на термин «канал» и позволяет двум соседям, подключенным к одному каналу, обмениваться пакетами, даже если они находятся в разных подсетях IPv6. Другими словами, интерфейс OSPF теперь подключается к каналу, а не к IP-подсети. Это изменение влияет на получение пакетов протокола OSPF, Hello-пакетов и содержимое Network-LSA.

2. **Удалена адресная семантика.** В OSPFv3 адресная семантика удалена из пакетов протокола и основных типов LSA. IPv6-адреса отсутствуют в пакетах OSPF за исключением содержимого пакетов Link State Update. Router-LSA и Network-LSA не содержат IP-адреса, а только описывают топологию. При этом размер идентификаторов Router ID, Area ID и LSA Link State ID остается прежним (32 бита) и записываются они в точечно-десятичной нотации (аналогично адресу IPv4), как в OSPFv2.

3. **Идентификация маршрутизаторов только по Router ID.** В OSPFv2 соседние маршрутизаторы на линиях связи «точка-точка» и виртуальных каналах идентифицируются с помощью Router ID, а в широковещательных сетях и в сетях NBMA — по IPv4-адресам интерфейсов. В OSPFv3 эта несогласованность удалена, и соседние маршрутизаторы всегда идентифицируются с помощью Router ID. Router ID, равный 0.0.0.0, зарезервирован и больше не должен использоваться.

4. **Добавлен диапазон рассылки.** В OSPFv3 диапазоны рассылки LSA обобщены и точно кодируются в поле LS type. Существуют три диапазона рассылки LSA в OSPFv3:

— *Link-local scope*. LSA рассыпается только в пределах локального канала. Для этого используется новый тип LSA — Link-LSA.

— *Area scope*. LSA рассыпается в пределах одной области OSPF. Используется для рассылки Router-LSA, Network-LSA, Inter-area-prefix-LSA, Inter-area-router-LSA и Intra-area-prefix-LSA.

— *AS scope*. LSA рассыпается через домен маршрутизации. Используется для AS-external-LSA.

5. **Использование адресов Link-Local IPv6 Unicast.** Адреса Link-Local IPv6 Unicast предназначены для взаимодействия внутри сегмента сети или по каналу связи «точка-точка» и используются только в пределах данного канала для обнаружения соседей, автоконфигурации и т. д. Маршрутизаторы не передают пакеты с адресами Link-Local Unicast, указанными в качестве источника или назначения, через другие линии связи. Адрес начинается с глобального префикса маршрутизации FE80::/10. Протокол OSPFv3 использует адреса Link-Local Unicast физических интерфейсов маршрутизаторов в качестве адресов источников пакетов и адресов следующих шагов в процессе передачи пакетов. Адреса Link-Local Unicast могут присутствовать только в Link-LSA. В LSA других типов их присутствие недопустимо.

6. **Поддержка множества копий протокола OSPF на канале.** Добавлена поддержка нескольких копий протокола OSPF на одном канале. В некоторых случаях множество маршрутизаторов OSPF могут быть подключены к общему широковещательному каналу, но не должны устанавливать соседство друг с другом. Например, четыре маршрутизатора подключены к одному каналу Ethernet. Маршрутизаторы 1 и 2 принадлежат одному домену OSPF, а маршрутизаторы 3 и 4 — другому. Соседство должно быть установлено между маршрутизаторами 1 и 2, а также маршрутизаторами 3 и 4. В OSPFv2 это можно сделать, например, с помощью аутентификации. OSPFv3 предлагает более простой способ. Он позволяет разным копиям OSPF, работающим на одном канале, добавлять в заголовок пакета OSPF идентификатор Instance ID, с помощью которого можно отличить одну копию от другой. Интерфейс, которому назначен определенный Instance ID, будет отбрасывать пакеты, чей Instance ID не соответствует его собственному.

7. **Из протокола удалена аутентификация.** В OSPFv3 полностью удалена аутентификация. Поля *AuthType* и *Authentication* удалены из заголовка пакета OSPF. Также все поля, относящиеся к аутентификации, удалены из структуры области и данных интерфейса. Для выполнения аутентификации OSPFv3 полагается на расширенные заголовки Authentication Header (AH) и Encapsulating Security Payload (ESP) IPv6.

8. **Обработка неизвестных типов LSA.** OSPFv2 всегда отбрасывает неизвестные типы LSA. В OSPFv3 обработка LSA сделана более гибкой. На основе поля LS type неизвестный тип LSA или может трактоваться как имеющий

локальный диапазон рассылки (Link-local scope), или запоминаться и рассылаться так, как будто тип известен. Тип обработки кодируется битом LSA Handling в поле LS type.

7.6.1. Пакеты OSPFv3

Протокол OSPF выполняется непосредственно поверх IPv6, в поле *Next Header* которого указывается номер 89.

```
Frame: Number = 1, Captured Frame Length = 90, MediaType = ETHERNET
└Ethernet: Etype = IPv6, DestinationAddress:[33-33-00-00-00-05], SourceAddress:[C2-00-1F-FA-00-01]
└Ipv6: Next Protocol = OSPF/MOSPF, Payload Length = 36
  └Versions: IPv6, Internet Protocol, DSCP 56
    -PayloadLength: 36 (0x24)
    -NextProtocol: OSPF/MOSPF, 89(0x59)
    -HopLimit: 1 (0x1)
    -SourceAddress: FE80:0:0:0:0:1
    -DestinationAddress: FF02:0:0:0:0:5
└Ospf: MessageType = Hello
  -Version: 3 (0x3)
  -MessageType: Hello, 1(0x01)
  -PacketLength: 36 (0x24)
  -RouterID: 1.1.1.1
  -AreaID: 0.0.0.1
  -Checksum: 64390 (0xFB86)
  -InstanceID: 0 (0x0)
  -Reserved: 0 (0x0)
  └HelloV3:
    -InterfaceID: 5 (0x5)
    -RouterPriority: 1 (0x1)
    └Options: R E V6
      -HelloInterval: 10 seconds (0xa)
      -RouterDeadInterval: 40 seconds (0x28)
      -DesignatedRouterID: 0.0.0.0
      -BackupDesignatedRouterID: 0.0.0.0
```

Рис. 7.50. Пакет OSPFv3

Протокол OSPFv3 использует те же пять типов пакетов (рис. 7.50), что и OSPFv2, и нумерует их так же: Type 1 — Hello, Type 2 — Database Description, Type 3 — Link State Request (LSR), Type 4 — Link State Update (LSU), Type 5 — Link State Acknowledgment (LSAck).

В формате заголовка OSPFv3 по сравнению с OSPFv2 произошли некоторые изменения (рис. 7.51):

- номер версии изменился с 2 на 3;
- удалены поля *AuType* и *Authentication*;
- добавлено поле *Instance ID* (*идентификатор копии*), которое позволяет нескольким копиям протокола OSPF существовать на одном канале.

Помимо формата заголовка, произошли изменения в форматах сообщений Hello и Database Description. Поле *Options* (*Опции*) в этих пакетах расширено до 24 бит. Из пакета Hello удалена адресная информация и добавлено поле *Interface ID* (*Идентификатор интерфейса*), которое используется для уникальной идентификации интерфейса маршрутизатора. Если

Биты: 0	7 8	15 16	
Версия (Version)	Тип (Type)	Длина пакета (Packet length)	
Идентификатор маршрутизатора (Router ID)			
Идентификатор области (Area ID)			
Контрольная сумма (Checksum)	Идентификатор копии (Instance ID)	Зарезервировано	

Рис. 7.51. Формат заголовка пакета OSPFv3

маршрутизатор становится Designated Router на канале, этот Interface ID будет указан в поле *Link State ID* Network-LSA.

7.6.2. Обзор LSA OSPFv3

По сравнению с OSPFv2 заголовок LSA OSPFv3 (рис. 7.52) изменился следующим образом:

- удалена адресная семантика;
- удалено поле *Options (Опции)*;
- увеличен размер поля *LS type (Тип LS)* до 16 бит. Теперь оно включает три предшествующих бита, кодирующих диапазон рассылки и способ обработки LSA неизвестных типов.

Биты: 0	15 16	
Возраст LS (LS age)	Тип LS (LS type)	
Идентификатор состояния канала (Link State ID)		
Объявляющий маршрутизатор (Advertising Router)		
Порядковый номер LS (LS sequence number)		
Контрольная сумма (LS checksum)	Длина (Length)	

Рис. 7.52. Заголовок LSA OSPFv3

Адреса в LSA OSPFv3 выражаются как *префикс/длина префикса*. Маршрут по умолчанию выражается префиксом с длиной 0.

Из Router-LSA и Network-LSA удалена вся адресная информация. Информация об интерфейсе маршрутизатора может быть распределена между несколькими Router-LSA. Поэтому получатель при выполнении вычислений по алгоритму Dijkstra должен объединять все Router-LSA, сгенерированные данным маршрутизатором.

Появился новый тип LSA, называемый *Link-LSA* (рис. 7.53). Link-LSA рассыпается только в пределах сегмента сети и генерируется маршрутизатором в том случае, если к каналу подключены два и более устройств. Оно служит для информирования других маршрутизаторов, подключенных к данному

сегменту сети об адресе Link-Local Unicast маршрутизатора, сгенерировавшего LSA, списке префиксов IPv6, ассоциированных с данным каналом, а также позволяет маршрутизатору объявлять о наборе опциональных битов в Network-LSA, генерируемых DR для этого канала. Другими словами, Link-LSA нужны для эффективного распространения информации об адресах Link-Local Unicast, которые используются в качестве адресов следующих шагов (next hop) в процессе создания таблицы маршрутизации OSPF.

```
⊕ Ipv6: Next Protocol = OSPF/MOSPF, Payload Length = 76
⊖ Ospf: MessageType = Link state update
    |- Version: 3 (0x3)
    |- MessageType: Link state update, 4 (0x04)
    |- PacketLength: 76 (0x4C)
    |- RouterID: 2.2.2.2
    |- AreaID: 0.0.0.1
    |- Checksum: 54175 (0xD39F)
    |- InstanceID: 0 (0x0)
    |- Reserved: 0 (0x0)
⊖ UpdateV3:
    |- LSANumbers: 1 (0x1)
    ⊖ LSA: Link-LSA
        |- LSAge: 1 seconds(0x1)
        ⊖ LSType: Link-LSA
        |- LinkStateID: 0.0.0.5
        |- AdvertisingRouter: 2.2.2.2
        |- LSSequenceNumber: 2147483650 (0x80000002)
        |- LSChecksum: 13579 (0x350B)
        |- Length: 56 (0x38)
    ⊖ LinkLSA:
        |- RouterPriority: 1 (0x1)
        ⊖ Options: DC R E V6
        |- LinkLocalInterfaceAddress: FE80:0:0:0:0:0:2
        |- PrefixNumbers: 1 (0x1)
    ⊖ Prefix:
        |- Prefix: 0:0:0:0::/64
            |- PrefixLength: 64 (0x40)
            ⊖ PrefixOption:
            |- Reserved: 0 (0x0)
            |- IPv6Prefix: 0:0:0:0::/64
```

Рис. 7.53. Link-LSA

В OSPFv3 появился новый тип LSA, называемый *Intra-area-prefix-LSA*. Он распространяет информацию о префиксе IPv6, которая в OSPFv2 включалась в Router-LSA и Network-LSA. Intra-area-prefix-LSA распространяется в пределах области и может выполнять одну из двух функций: связывать

список префиксов IPv6 с каналом транзитной сети, ссылаясь на Network-LSA, или связывать список префиксов IPv6 с маршрутизатором, ссылаясь на Router-LSA.

Маршрутизатор может генерировать множество Intra-area-prefix-LSA (рис. 7.54), каждое из которых должно иметь уникальный идентификатор Link State ID.

Два типа LSA переименованы следующим образом:

- Type-3 Summary-LSA переименована в *Inter-area-prefix-LSA*;
- Type-4 Summary LSA переименована в *Inter-area-router-LSA*.

```
▫ LSA: Intra-Area-Prefix-LSA
  - LSAge: 1 seconds (0x1)
  - LSType: Intra-Area-Prefix-LSA
  - LinkStateID: 0.0.20.0
  - AdvertisingRouter: 1.1.1.1
  - LSSequenceNumber: 2147483649 (0x80000001)
  - LSChecksum: 36636 (0x8F1C)
  - Length: 44 (0x2C)
  □ IntraAreaPrefixLSA:
    - PrefixNumbers: 1 (0x1)
    - ReferencedLSType: Network-LSA
      ReferencedLinkStateID: 0.0.0.5
      ReferencedAdvertisingRouter: 1.1.1.1
      □ OSPFIPv6Prefix: 2001:DB8:0:12::/64 Metric:0
  ▫ LSA: Intra-Area-Prefix-LSA
    - LSAge: 3600 seconds (0xE10)
    - LSType: Intra-Area-Prefix-LSA
    - LinkStateID: 0.0.0.0
    - AdvertisingRouter: 1.1.1.1
    - LSSequenceNumber: 2147483650 (0x80000002)
    - LSChecksum: 5366 (0x14F6)
    - Length: 32 (0x20)
    □ IntraAreaPrefixLSA:
      - PrefixNumbers: 0 (0x0)
      - ReferencedLSType: Router-LSA
        ReferencedLinkStateID: 0.0.0.0
        ReferencedAdvertisingRouter: 1.1.1.1
```

Рис. 7.54. Intra-area-prefix-LSA

8. Протоколы транспортного уровня

Основным протоколом сетевого уровня является протокол IP. Он передает сетевые пакеты *без установления соединения, без обеспечения надежности и без подтверждения доставки*. Получается, что при организации передачи данных на основе протокола IP, отправитель не будет знать, доставляются ли его IP-пакеты получателю или нет.

Во время передачи по сети IP-пакеты с определенной долей вероятности могут быть искажены или потеряны. Несмотря на то что некоторые сетевые приложения организуют собственную проверку доставки передаваемых данных, а также имеют собственные средства диагностики и обработки вероятных ошибок, существуют и такие приложения, которые перекладывают выполнение этих функций на стандартные сетевые протоколы. Обеспечить сетевым приложениям передачу данных с той степенью надежности, которая им требуется, способны протоколы *транспортного уровня*.

На транспортном уровне функционируют два основных протокола — TCP (Transmission Control Protocol, протокол управления передачей) и UDP (User Datagram Protocol, протокол дейтаграмм пользователей).

Протокол TCP обеспечивает установку соединения между отправителем и получателем, разбиение крупных информационных блоков на сегменты ограниченной длины, а также их гарантированную доставку получателю в заданном порядке и без ошибок. Функционирование протокола TCP предполагает его взаимодействие с протоколами прикладного уровня, например, HTTP, FTP, SMTP и др.

Протокол UDP в отличие от TCP не устанавливает соединение перед передачей данных и не требует от получателя подтверждений о доставке, но за счет именно этих особенностей он работает быстрее, чем TCP. Протокол UDP используют в тех задачах, где в первую очередь необходимо обеспечить хорошую скорость передачи данных, а гарантия доставки и надежность имеют второстепенное значение. Примером такой задачи является передача потокового видео по технологии IPTV.

8.1. Адресация протоколов TCP и UDP

На сетевом уровне модели OSI для уникальной идентификации сетевого интерфейса используется сетевой адрес, например адрес IPv4. Сетевая адресация является механизмом, с помощью которого пакеты маршрутизируются в нужную сеть. На транспортном уровне существует еще один уровень адресации, который обеспечивает возможность приема/передачи данных несколькими сетевыми приложениями, одновременно работающими на одном IP-интерфейсе. Протоколы транспортного уровня TCP и UDP используют концепцию *порта и сокета*.

Большинство передач с использованием протоколов стека TCP/IP имеют форму обмена информацией между программой на одном устройстве с соответствующей программой на другом устройстве. Программа, которая

выполняется в текущий момент, называется процессом. Прикладной процесс TCP/IP — это часть сетевого программного обеспечения, которое отправляет и получает информацию с помощью протоколов стека TCP/IP.

На узле с поддержкой стека TCP/IP обычно бывает запущено множество процессов, которые отправляют и получают данные. Например, на компьютере могут одновременно работать Web-браузер, Skype и сетевая игра. Все эти процессы должны отправлять и получать данные через один и тот же интерфейс, используя протокол IP. Данные от сетевых приложений поступают на транспортный уровень, где они обрабатываются протоколом TCP или UDP. Далее они передаются на сетевой уровень, где упаковываются в IP-пакеты и отправляются в соответствующие пункты назначения. Передача данных от разных прикладных процессов по одному соединению называется *мультиплексированием* (multiplexing).

Одновременно с отправкой данных разных прикладных процессов соответствующим получателям протокол IP получает множество пакетов, предназначенных для разных прикладных процессов, работающих на этом

устройстве. Он пересыпает поток не связанных друг с другом данных на транспортный уровень, который обрабатывает их и далее передает соответствующим процессам. Процесс, обратный мультиплексированию, называется *демультиплексированием* (demultiplexing) (рис. 8.1).

Сетевые приложения используют *клиент/серверную* модель работы: *клиентский процесс* обычно запускается на клиентском устройстве и инициирует взаимодействие с сервером с целью

Рис. 8.1. Демультиплексирование на основе портов

выполнения какой-либо функции; *серверный процесс*, обычно запускаемый на аппаратном сервере, принимает запросы клиентов и отвечает на них. Классическим примером этого является запрос Web-страницы с Web-сервера с помощью протокола HTTP. Web-браузер является HTTP-клиентом, который отправляет запросы на HTTP (Web)-сервер.

Когда сетевой уровень получает IP-пакет, содержащий данные для транспортного уровня, он должен передать его дальше. Для того чтобы определить протокол-получатель транспортного уровня, необходимо проверить поле Protocol (IPv4) или Next Header (IPv6). В этих полях содержится код, идентифицирующий протокол, которому предназначены данные. Поскольку большинство приложений используют либо протокол TCP, либо UDP, полученный на сетевом уровне пакет передается одному из этих протоколов. Протоколы TCP и UDP могут одновременно использоваться множеством прикладных процессов, и поэтому они должны уметь отличать один прикладной процесс от другого. Для этого используется адрес транспортного уровня, называемый *порт* (port).

8. Протоколы транспортного уровня

Порт — это 16-битный номер, служащий для идентификации протокола высокого уровня или прикладного процесса (программы), которому должны быть доставлены входящие данные. Номера портов могут принимать значения от 0 до 65535. Протоколы TCP и UDP используют одинаковый диапазон номеров, но при этом они независимы друг от друга.

В форматах сообщений TCP и UDP присутствуют по два адресных поля: *Source Port* и *Destination Port*. Они идентифицируют прикладные процессы на машине отправителя и на машине получателя.

Централизованно управляет адресным пространством номеров портов IANA. Спектр номеров портов разделен на три диапазона:

- *Общеизвестные порты* (Well Known Port), диапазон от 0 до 1023. Этот диапазон номеров назначается IANA, он зарезервирован для стандартных протоколов (табл. 8.1). В большинстве случаев эти номера портов используются серверами, такими как Web-сервер, FTP-сервер и подобными. По этой причине их иногда называют *системными портами* (System Port). Общеизвестные порты позволяют клиентам найти серверы без дополнительной конфигурационной информации;

- *Зарегистрированные порты* (Registered Port), диапазон от 1024 до 49 151 (табл. 8.2). Эти номера портов зарезервированы для использования серверными приложениями, которые не стандартизированы. Для того чтобы такие приложения не конфликтовали друг с другом, IANA назначает им номера из этого диапазона. Например, кто-то создал серверное приложение и обращается в IANA с просьбой зарегистрировать один из номеров для него. После утверждения IANA регистрирует номер порта и присваивает его этому приложению. Поскольку номера из этого диапазона запрашиваются пользователями, они также иногда называются *пользовательскими* (User Port);

- *Динамические* (Dynamic)/*частные* (Private) или *эфемерные* (Ephemeral) порты, диапазон от 49 152 до 65 535. Эти номера портов не назначаются и не регистрируются IANA.

Таблица 8.1. **Общеизвестные номера портов TCP и UDP**

Номер порта	Транспортный протокол	Сервис
20	TCP	File Transfer Protocol (FTP) — данные
21	TCP	File Transfer Protocol (FTP) — управление
22	TCP, UDP	Secure Shell (SSH)
23	TCP, UDP	Telnet
25	TCP	Simple Mail Transfer Protocol (SMTP)
49	TCP, UDP	Login Host Protocol (TACACS)
53	TCP, UDP	Domain Name Server
67	UDP	Bootstrap Protocol/DHCPv4 Server
68	UDP	Bootstrap Protocol/DHCPv4 Client

Окончание табл. 8.1

Номер порта	Транспортный протокол	Сервис
69	UDP	Trivial File Transfer Protocol (TFTP)
80	TCP, UDP	HyperText Transfer Protocol (HTTP)
109	TCP, UDP	Post Office Protocol — Version 2 (POP2)
110	TCP, UDP	Post Office Protocol — Version 3 (POP3)
123	TCP, UDP	Network Time Protocol (NTP)
161	TCP, UDP	Simple Network Management Protocol (SNMP)
389	TCP, UDP	Lightweight Directory Access Protocol (LDAP)
443	TCP, UDP	HTTP protocol over TLS/SSL (HTTPS)
500	UDP	IPSec Internet Key Exchange (IKE)
520	UDP	Routing Information Protocol (RIP)
521	UDP	Routing Information Protocol Next Generation (RIPng)
546	TCP, UDP	DHCPv6 Client
547	TCP, UDP	DHCPv6 Server
989	TCP, UDP	FTP over TLS/SSL (FTPS) — данные
990	TCP, UDP	FTP over TLS/SSL (FTPS) — управление

Таблица 8.2. Зарегистрированные номера портов TCP и UDP

Номер порта	Транспортный протокол	Сервис
1701	TCP, UDP	Layer Two Tunneling Protocol (L2TP)
1723	TCP, UDP	Point-To-Point Tunneling Protocol (PPTP)
2049	TCP, UDP	Network File System (NFS)

Общеизвестные и зарегистрированные номера портов требуются серверным процессам, так как клиент должен их знать, чтобы начать процесс передачи данных с использованием TCP или UDP. Когда клиент отправляет запрос, в поле *Destination Port* блока данных TCP или UDP он указывает общеизвестный или зарегистрированный номер порта приложения, которому предназначены данные.

Сервер в отличие от клиента не инициирует начало передачи данных. При отправке ответа клиенту он не должен указывать общеизвестные или зарегистрированные номера портов. Другими словами, клиентские процессы не могут использовать общеизвестные или зарегистрированные номера портов. Причиной этого является то, что на каком-нибудь устройстве одновременно могут быть запущены клиентское и серверное приложение, использующее одинаковый сетевой протокол. Если сервер получит, например, HTTP-запрос на порт 80 от клиентского процесса такого устройства

8. Протоколы транспортного уровня

и отправит ему ответ на порт 80, то ответ получит не клиентское приложение, инициировавшее запрос, а серверное (рис. 8.2).

Для того чтобы отправить ответ клиентскому приложению, сервер должен знать, какой номер порта оно использует. Каждый раз, когда клиентский процесс инициирует передачу данных по TCP или UDP, ему динамически назначается временный или *эфемерный номер* порта, который будет использоваться для данного обмена. Номер порта выбирается псевдослучайным образом из диапазона динамических портов. Таким образом, когда клиентский процесс отправляет запрос серверу, в поле *Source Port* блока данных TCP или UDP он указывает свой динамически назначенный номер порта, а в поле *Destination Port* — общезвестный или зарегистрированный номер порта серверного процесса. При отправлении ответа клиентскому приложению серверное приложение в поле *Source Port* указывает свой общезвестный или зарегистрированный номер порта, в поле *Destination Port* — номер порта из запроса клиента (рис. 8.3).

```
Frame: Number = 871, Captured Frame Length = 116, MediaType = WiFi
@ WiFi: [Unencrypted Data] .T....., (I)
@ LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP(Sub-Network Access Protocol), DSAP = SNAP(Sub-Network Access Protocol)
@ Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION
@ IPv4: Src = 192.168.100.4, Dest = 216.92.67.219, Next Protocol = TCP, Packet ID = 9741, Total IP Length = 52
@ Tcp: Flags=.....S., SrcPort=49444, DstPort=HTTP(80), PayloadLen=0, Seq=4175484916, Ack=0, Win=8192 ( Negotiating scale factor
  -SrcPort: 49444
  -DstPort: HTTP(80)
  -SequenceNumber: 4175484916 (0xF8E0D7F4)
  -AcknowledgementNumber: 0 (0x0)
  -DataOffset: 128 (0x80)
  -Flags: .....S.
  -Window: 8192 ( Negotiating scale factor 0x2 ) = 8192
  -Checksum: 0x7B57, Disregarded
  -UrgentPointer: 0 (0x0)
  @ TCPOptions:
```

Рис. 8.2. Запрос, отправленный клиентом Web-серверу

```
Frame: Number = 881, Captured Frame Length = 116, MediaType = WiFi
@ WiFi: [Unencrypted Data] F.....P, (I) RSSI = -53 dBm, Rate = 0.5 Mbps
@ LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP(Sub-Network Access Protocol), DSAP = SNAP(Sub-Network Access Protocol)
@ Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION
@ IPv4: Src = 216.92.67.219, Dest = 192.168.100.4, Next Protocol = TCP, Packet ID = 41860, Total IP Length = 52
@ Tcp: Flags=...A..S., SrcPort=HTTP(80), DstPort=49444, PayloadLen=0, Seq=3612665421, Ack=4175484917, Win=65535 ( Negotiated sca
  -SrcPort: HTTP(80)
  -DstPort: 49444
  -SequenceNumber: 3612665421 (0xD754E64D)
  -AcknowledgementNumber: 4175484917 (0xF8E0D7F5)
  -DataOffset: 128 (0x80)
  -Flags: ...A..S.
  -Window: 65535 ( Negotiated scale factor 0x6 ) = 4194240
  -Checksum: 0xDP60, Good
  -UrgentPointer: 0 (0x0)
  @ TCPOptions:
```

Рис. 8.3. Ответ Web-сервера клиенту

Однако из этого правила есть исключения. Для протоколов DHCPv4 и DHCPv6 определены общезвестные номера портов как для серверных, так и для клиентских процессов.

Для однозначной идентификации прикладного процесса TCP/IP на устройстве используется комбинация IP-адреса интерфейса, через который

идет обмен данными, и номера порта транспортного протокола. Этот комбинированный адрес называется *сокет* (socket). Для указания сокета используется следующая нотация:

<IP Address>:<Port Number>

Например, если HTTP-сервер работает по адресу 192.168.90.90, соответствующий ему сокет на этом сайте будет 192.168.90.90:80.

Иногда вместо IP-адреса может использоваться имя узла (host name). В этом случае нотация будет следующей:

<Host Name>:<Port Number>

Сокет является фундаментальной концепцией работы сетевого программного обеспечения TCP/IP. Он служит основой для программного интерфейса (API) с аналогичным названием. Версия API для Windows называется Windows Sockets или WinSock. Эти API позволяют приложениям использовать TCP/IP для коммуникаций.

Обмен данными между парой устройств состоит из серии сообщений, отправляемых с сокета одного устройства на сокет другого. В один момент времени на каждом устройстве может быть множество активных TCP-подключений. Эти подключения должны обслуживаться, и поэтому они требуют уникальной идентификации. Каждое соединение уникально идентифицируется с помощью комбинации сокетов двух взаимодействующих процессов — локального и удаленного:

<local IP Address>:<local Port Number>,
<remote IP Address>:<remote Port Number>

В отличие от TCP протокол UDP не устанавливает соединения. Тем не менее пара сокетов может использоваться для идентификации взаимодействующих по UDP процессов, но не имеет такого большого значения как, в TCP.

8.2. Протокол UDP

User Datagram Protocol (UDP) является протоколом транспортного уровня стека TCP/IP, который определен в RFC 768. Он был разработан для использования протоколами прикладного уровня, которым не требуются надежность, управление потоком и обнаружение ошибок. UDP — простой и быстрый протокол, который выполняет функции мультиплексирования/демультиплексирования при отправлении и получении дейтаграмм, обеспечивая адресацию транспортного уровня в форме UDP-портов. Другими словами, UDP предоставляет механизм, с помощью которого одно приложение отправляет дейтаграммы другому с минимальным количеством операций. При отправлении эти операции включают получение данных от вышележащего уровня, инкапсуляцию их в UDP-сообщение, передачу UDP-сообщения на уровень IP. При получении данных порядок операций будет обратным. Поскольку количество операций UDP минимально, функции упорядоченной надежной доставки данных возлагаются на приложение.

8. Протоколы транспортного уровня

8.2.1. Формат дейтаграммы UDP

Формат дейтаграммы UDP очень простой. Он включает 8-байтный заголовок и поле данных, как показано на рис. 8.4.



Рис. 8.4. Формат UDP-дейтаграммы

В заголовке дейтаграммы присутствуют следующие поля:

- *Порт отправителя (Source Port)*. Поле длиной 16 бит, которое содержит номер порта процесса, породившего сообщение. Это порт, на который будет адресоваться ответ;
- *Порт получателя (Destination Port)*. Поле длиной 16 бит, содержащее номер порта процесса, которому предназначены данные на узле-получателе;
- *Длина (Length)*. Длина дейтаграммы в байтах, включая заголовок и данные;
- *Контрольная сумма (Checksum)*. Опциональное поле длиной 16 бит, вычисленное на основе суммы псевдозаголовка, заголовка UDP и данных. Псевдозаголовок (рис. 8.5) помещается перед заголовком UDP и содержит IP-адреса источника и назначения, номер протокола и длину UDP. Эта

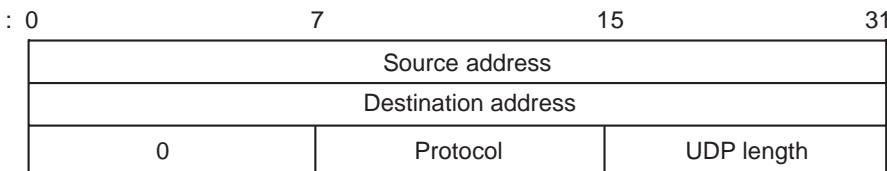


Рис. 8.5. Псевдозаголовок

информация обеспечивает защиту от сообщений, ошибочно отправленных по другому маршруту. Псевдозаголовок используется только для вычисления контрольной суммы и не передается по сети.

8.3. Протокол TCP

Transmission Control Protocol (TCP) является протоколом транспортного уровня стека TCP/IP. В отличие от UDP он предоставляет приложениям больше возможностей: выполняет установление соединения, обеспечивает надежность, обнаружение ошибок и управление потоком. TCP позволяет

приложениям отправлять данные как поток байтов и автоматически упаковывает их в сегменты соответствующего размера.

Протокол TCP был описан в RFC 793 в сентябре 1981 года. Со временем он был усовершенствован, были исправлены ошибки и неточности. На сегодняшний день существует множество RFC, являющихся дополнениями к RFC 793. Для удобной работы с ними был создан специальный указатель, описанный в RFC 7414 «A Roadmap for Transmission Control Protocol (TCP)».

TCP обладает следующими характеристиками:

- *С установлением соединения*: TCP требует, чтобы устройства перед началом передачи данных установили соединение друг с другом;

- *Двунаправленный*: после установления соединения данные между устройствами передаются в обоих направлениях, т. е. оба устройства могут отправлять и получать данные, несмотря на то, какое из них инициировало установку соединения.

- *С поддержкой мультиплексирования*: соединения TCP между процессами на разных устройствах идентифицируются парой сокетов. Это позволяет устройству иметь множество открытых TCP-соединений, каждое из которых обслуживается независимо друг от друга. Один сокет может использоваться одновременно для нескольких соединений. Другими словами, два и более соединения могут оканчиваться одним сокетом.

- *С потоковой передачей данных*: большинство низкоуровневых протоколов разработаны так, что для их использования высокоуровневые протоколы должны отправлять им данные, разбитые на блоки (дейтаграммы, кадры). TCP позволяет приложениям отправлять ему непрерывный поток данных для передачи по сети. Приложениям не надо нарезать данные на блоки или дейтаграммы. Это делает TCP, группируя байты в сегменты, которые передаются на IP-уровень для отправки получателю.

- *Надежный*: TCP отслеживает отправленные и полученные данные. Для этого в каждом передаваемом сегменте содержится порядковый номер первого байта данных. Получение сегмента должно быть подтверждено. Если в течение определенного времени подтверждение не получено, данные передаются повторно. Получатель использует порядковый номер для упорядочения сегментов и исключения их дублирования.

- *Управление потоком*: TCP управляет потоком передаваемых данных. Когда получатель посылает отправителю подтверждение приема данных, он сообщает о количестве байтов, которое он может принять, чтобы избежать переполнения его внутренних буферов.

8.3.1. Сегмент TCP

Отправляющая и принимающая TCP-подсистемы обмениваются данными в виде *сегментов* (segment), на которые разбивается непрерывный поток байтов, поступающий от приложений.

Сегменты разработаны таким образом, что могут одновременно переносить управляющую информацию и данные. Это позволяет использовать их для разных целей.

8. Протоколы транспортного уровня

Сегмент TCP состоит из фиксированного 20-байтового заголовка (плюс необязательная часть), за которым могут следовать байты данных (рис. 8.6).

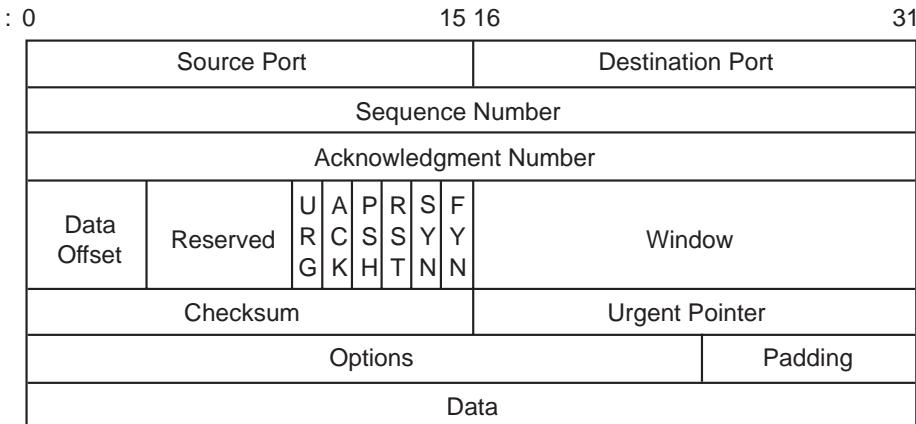


Рис. 8.6. Формат сегмента TCP

Заголовок сегмента включает 12 полей.

- *Порт источника (Source Port)*: поле длиной 16 бит, содержащее номер порта процесса-источника.
- *Порт назначения (Destination Port)*: поле длиной 16 бит, содержащее номер порта процесса-получателя.
- *Порядковый номер (Sequence Number)*: поле длиной 32 бита, содержащее порядковый номер первого байта данных этого сегмента. Если установлен управляющий бит SYN, в поле указан первоначальный порядковый номер (Initial Sequence Number, ISN), а первый байт данных получает номер ISN+1.
- *Номер подтверждения (Acknowledgment Number)*: поле длиной 32 бита. Если установлен управляющий бит ACK, поле содержит значение следующего порядкового номера байта, который получатель ожидает получить от отправителя.
- *Смещение данных (Data Offset)*: поле длиной 4 бита, которое указывает количество 32-битных слов в заголовке TCP. Оно показывает, где начинаются данные.
- *Зарезервировано (Reserved)*: поле длиной 6 бит. Зарезервировано для использования в будущем. Должно содержать 0.
- *Управляющие биты (Control Bits)*: поле содержит 6 бит, используемых для передачи управляющей информации:
 - **URG**: бит Urgent. Показывает, что поле *Urgent Pointer* имеет значения для этого сегмента.

- **ACK:** бит Acknowledgment. Показывает, что этот сегмент содержит подтверждение, и поле *Acknowledgment Number* имеет значения для этого сегмента.
- **PSH:** бит Push. Показывает, что отправитель сегмента использует функцию *Push*, с помощью которой он просит получателя доставить данные приложению сразу по получении пакета, а не хранить их в буфере, пока тот не наполнится.
- **RST:** бит Reset. Используется для сброса соединения в случае сбоя, отказа от неверного сегмента или от попытки создать соединение.
- **SYN:** бит Synchronize. Используется для установки соединения.
- **FIN:** бит Finish. Используется для разрыва соединения. Он указывает на то, что у отправителя больше нет данных для передачи.
- **Размер окна (Window):** поле длиной 16 бит. Содержит количество байтов данных, начиная с байта, указанного в поле *Acknowledgment Number*, которое отправитель этого сегмента готов принять за один раз.
- **Контрольная сумма (Checksum):** поле длиной 16 бит. Контрольная сумма, вычисленная на основе суммы псевдоголовки, заголовка TCP и данных служит для повышения надежности. Псевдоголовок аналогичен используемому UDP, с тем отличием, что номер протокола TCP равен 6. Контрольная сумма является обязательной.
- **Указатель срочности (Urgent Pointer):** поле длиной 16 бит. Это поле имеет значение, если установлен контрольный бит URG. Оно содержит смещение в байтах от текущего порядкового номера и указывает на порядковый номер первого байта данных, следующего за срочными данными.
- **Опции (Options):** поле переменной длины. Позволяет включать в TCP-сегмент различные опциональные параметры. Существует множество опций. Опция может быть длиной 1 байт или переменной длины. Опция длиной 1 байт содержит только поле «*Tip*», опция переменной длины содержит три поля: «*Tip*», «Длина», «Значение».
- **Заполнение (Padding):** поле переменной длины. Служит для дополнения заголовка TCP нулями, чтобы его длина стала кратной 32 битам.

8.3.2. Модель управления TCP-соединением

Прежде чем перейти к рассмотрению процессов установления и разрыва TCP-соединения, кратко опишем конечный автомат TCP. Конечный автомат — это средство, используемое для описания работы алгоритма или протокола. Он определяет конечное число состояний, событий, которые могут вызвать переход из одного состояния в другое и действий, предпринимаемых в ответ на события.

Процесс установки и разрыва соединения TCP может быть представлен в виде конечного автомата, содержащего 11 событий: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT и фиктивного состояния CLOSED. Краткое описание состояний приведено в табл. 8.3.

8. Протоколы транспортного уровня

Таблица 8.3. Состояния конечного автомата TCP

Состояние	Описание
CLOSED	Закрыто. Соединение не является активным и не находится в процессе установления. Поэтому в RFC оно называется «фиктивным». Это состояние по умолчанию, которое предшествует процессу начала установки соединения
LISTEN	Ожидание. Устройство ожидает получения входящего запроса (SYN) от любого другого устройства
SYN-SENT	Запрос на соединение отправлен. Устройство отправило запрос на соединение (SYN) и ожидает получение соответствующего запроса от другого устройства
SYN-RECEIVED	Получен запрос на соединение. После того как устройство получило запрос на соединение (SYN) от другого устройства и отправило свой, оно ожидает подтверждение (ACK) для завершения установки соединения
ESTABLISHED	Установлено. Нормальное состояние передачи данных
CLOSE-WAIT	Ожидание закрытия. Устройство ожидает получение запроса на завершение соединения (FIN) от другого устройства
LAST-ACK	Последнее подтверждение. Устройство, которое получило запрос на завершение соединения (FIN) от другого устройства и подтвердило его, отправляет свой запрос на завершение соединения (FIN) и ожидает его подтверждения (ACK)
FIN-WAIT-1	Ожидание запроса на завершение соединения. Устройство ожидает получение запроса на завершение соединения (FIN) от другого устройства или подтверждения для ранее отправленного им запроса на завершение соединения (FIN)
FIN-WAIT-2	Ожидание запроса на завершение соединения. Устройство ожидает получение запроса на завершение соединения (FIN) от другого устройства
CLOSING	Закрытие. Устройство ожидает получение подтверждения для ранее отправленного им запроса на завершение соединения (FIN)
TIME-WAIT	Время ожидания. Устройство ожидает время, требуемое для получения другим устройством отправленного ему подтверждения запроса на завершение соединения (FIN). После истечения этого периода времени устройство переходит в состояние CLOSED

Соединение TCP переходит из одного состояния в другое в ответ на происходящие события. События — это пользовательские вызовы OPEN, SEND, RECEIVE, CLOSE, ABORT и STATUS; входящие сегменты, содержащие флаги SYN, ACK, RST и FIN; тайм-ауты. Диаграмма на рис. 8.7 иллюстрирует изменения состояний вместе с вызывающими их событиями и результатирующими действиями.

Каждое соединение начинается в состоянии *CLOSED* (закрыто). Оно может покинуть это состояние, предпринимая либо активную (*SYN-SENT*),

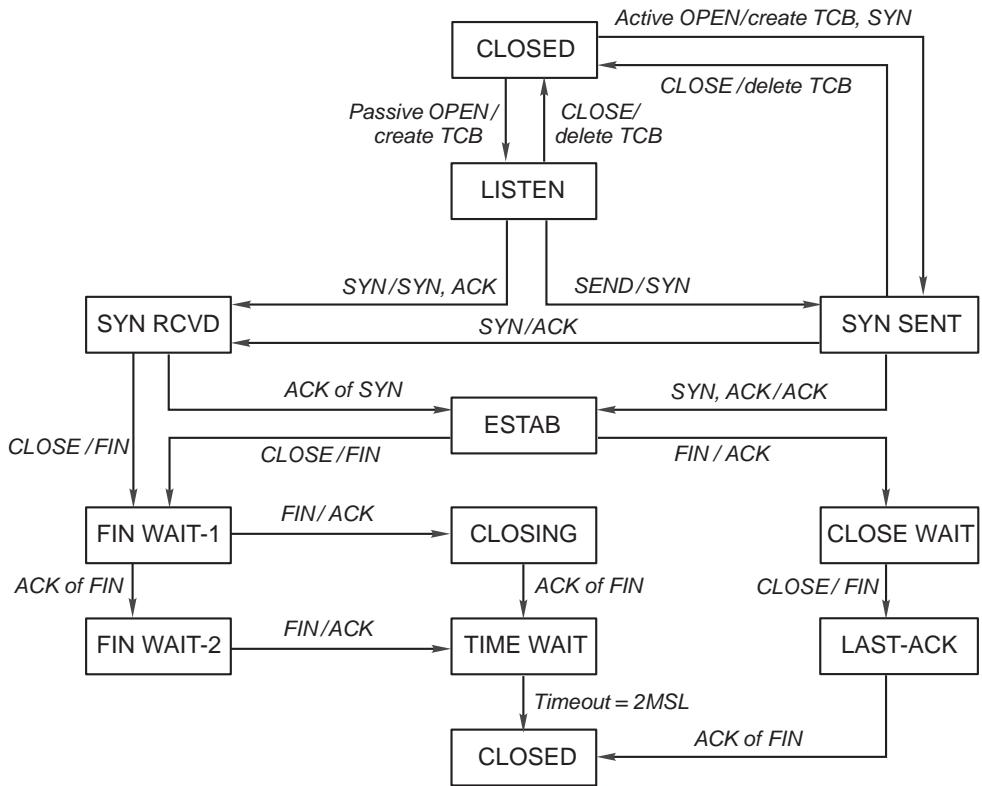


Рис. 8.7. Диаграмма состояний соединения TCP

либо пассивную (*LISTEN*) попытку открыть соединение. Если противоположная сторона осуществляет противоположные действия, соединение устанавливается и переходит в состояние *ESTABLISHED* (установлено). Инициатором разрыва соединения может выступить любая сторона. По завершении процесса разъединения соединение возвращается в состояние *CLOSED*.

Машина состояний используется для каждого соединения в отдельности. Это означает, что в любой момент времени TCP может находиться в одном

состоянии для подключения к сокету X, и в другом — для подключения к сокету Y. Также изменение состояний двух процессов на концах конкретного соединения не является симметричным, поскольку роли устройств не симметричны: одно устройство инициирует или завершает соединение, другое отвечает. При этом не исключается, что оба устройства могут инициировать или завершать соединение одновременно.

8.3.3. Установка соединения TCP

Каждая из сторон TCP-соединения готовится к началу его установки, выполняя одну из операций *OPEN*, которая бывает двух видов:

- *Active OPEN*: процесс (обычно клиентский), который инициирует соединение, отправляя сообщение о начале его установки (SYN) процессу (обычно серверному), с которым он хочет обмениваться данными;
- *Passive OPEN*: процесс (обычно серверный), который пассивен до тех пор, пока другой процесс, выполняющий операцию *Active OPEN*, не попытается подключиться к нему.

Поскольку соединения отличаются друг от друга, необходимо хранить данные о каждом из них. Для этой цели TCP использует специальную структуру данных, называемую *блоком управления передачей* (Transmission Control Block, TCB). Протокол TCB создается каждой из сторон TCP-соединения во время выполнения операции *OPEN*. TCB содержит такую информацию о соединении, как номера идентифицирующих его сокетов и указатели на буфера, в которых хранятся входящие и исходящие данные. TCB также используется для реализации механизма скользящего окна (sliding window). Он хранит переменные, которые отслеживают количество принятых и подтвержденных байтов, полученных и еще не подтвержденных байтов, текущий размер окна и т. д.

Блок управления передачей поддерживается на протяжении всего времени существования соединения и удаляется, когда соединение полностью завершено, и устройство возвращается в состояние *CLOSED*.

Процесс установки TCP-соединения называется *трехсторонним рукопожатием* (*three-way handshake*), так как состоит из трех шагов.

Инициатор соединения (например, клиент), выполнив операцию *Active OPEN* и создав TCB, переходит в состояние *SYN-SENT*. Он посыпает TCP-сегмент с установленным битом SYN (запрос на соединение) (рис. 8.8) и ждет подтверждения от получателя (например, сервера).

В процессе установки TCP-соединения устройства сообщают друг другу о порядковом номере, который они планируют использовать при первой передаче данных. Эта процедура называется *синхронизацией порядкового номера* (*sequence number synchronization*).

Начальный порядковый номер выбирается произвольным образом и помещается в поле *Sequence Number* TCP-сегмента с установленным битом SYN.

Технологии TCP/IP в современных компьютерных сетях

```
Frame: Number = 2568, Captured Frame Length = 116, MediaType = WiFi
WiFi: [Unencrypted Data] .T...., (I)
LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP(Sub-Network Access Protocol), DSAP = SNAP(Sub-Network Access Protocol)
Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION
Ipv4: Src = 192.168.100.4, Dest = 193.7.160.227, Next Protocol = TCP, Packet ID = 23407, Total IP Length = 52
Tcp: Flags=.....S., SrcPort=49729, DstPort=HTTP(80), PayloadLen=0, Seq=3691927533, Ack=0, Win=8192 ( Negotiating scale fact.
- SrcPort: 49729
- DstPort: HTTP(80)
- SequenceNumber: 3691927533 (0xDC0E57ED)
- AcknowledgementNumber: 0 (0x0)
@ DataOffset: 128 (0x80)
@ Flags: .....S.
- CWR: (0.....) CWR not significant
- ECE: (0.....) ECN-Echo not significant
- Urgent: (0.....) Not Urgent Data
- Ack: (....0...) Acknowledgement field not significant
- Push: (....0...) No Push Function
- Reset: (....0..) No Reset
- Syn: (.....1.) Synchronize sequence numbers
- Fin: (.....0) Not End of data
- Window: 8192 ( Negotiating scale factor 0x2 ) = 8192
- Checksum: 0xD1F0, Disregarded
- UrgentPointer: 0 (0x0)
@ TCPOptions:
```

Рис. 8.8. Запрос на соединение SYN

Поэтому клиент, инициируя соединение, выбирает с помощью генератора случайных чисел начальный порядковый номер и посыпает его в TCP-сегменте с битом SYN серверу. Сервер пассивно ожидает входящего соединения, выполнив операцию *Passive OPEN*, создав TCB и перейдя в состояние *LISTEN*. Когда отправленный сегмент прибывает в пункт назначения, TCP-подсистема проверяет, выполнил ли какой-нибудь процесс операцию *LISTEN*, указав в качестве параметра тот же порт, который содержится в поле *Порт назначения* сегмента. Если такого процесса нет, она отвечает отправкой сегмента с установленным битом RST для отказа от соединения.

Если сервер принимает соединение, он переходит в состояние *SYN-RECEIVED*. В ответ клиенту отправляется TCP-сегмент, с установленными битами SYN и ACK (SYN + ACK) (рис. 8.5).

```
Frame: Number = 2575, Captured Frame Length = 116, MediaType = WiFi
WiFi: [Unencrypted Data] F....P, (I) RSSI = -50 dBm, Rate = 0.5 Mbps
LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP(Sub-Network Access Protocol), DSAP = SNAP(Sub-Network Access Protocol)
Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION
Ipv4: Src = 193.7.160.227, Dest = 192.168.100.4, Next Protocol = TCP, Packet ID = 0, Total IP Length = 52
Tcp: Flags=...A..S., SrcPort=HTTP(80), DstPort=49729, PayloadLen=0, Seq=3467299494, Ack=3691927534, Win=29200 ( Negotiated
- SrcPort: HTTP(80)
- DstPort: 49729
- SequenceNumber: 3467299494 (0xCEAACAA6)
- AcknowledgementNumber: 3691927534 (0xDC0B57EB)
@ DataOffset: 128 (0x80)
@ Flags: ...A..S.
- CWR: (0.....) CWR not significant
- ECE: (0.....) ECN-Echo not significant
- Urgent: (0.....) Not Urgent Data
- Ack: (....1...) Acknowledgement field significant
- Push: (....0...) No Push Function
- Reset: (....0..) No Reset
- Syn: (.....1.) Synchronize sequence numbers
- Fin: (.....0) Not End of data
- Window: 29200 ( Negotiated scale factor 0x7 ) = 3737600
- Checksum: 0xE6C8, Good
- UrgentPointer: 0 (0x0)
@ TCPOptions:
```

Рис. 8.9. Подтверждение запроса на соединение и запрос на соединение (SYN+ACK)

8. Протоколы транспортного уровня

```

Frame: Number = 2576, Captured Frame Length = 104, MediaType = WiFi
@ WiFi: [Unencrypted Data] .T....., (I)
@ LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP(Sub-Network Access Protocol), DSAP = SNAP(Sub-Network Access Protocol)
@ Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION
@ Ipv4: Src = 192.168.100.4, Dest = 193.7.160.227, Next Protocol = TCP, Packet ID = 23408, Total IP Length = 40
@ Tcp: Flags=...A...., SrcPort=49729, DstPort=HTTP(80), PayloadLen=0, Seq=3691927534, Ack=3467299495, Win=16560 (scale factor
  SrcPort: 49729
  DstPort: HTTP(80)
  SequenceNumber: 3691927534 (0xDC0E57EE)
  AcknowledgementNumber: 3467299495 (0xCEAACAA7)
  DataOffset: 80 (0x50)
@ Flags: ...A....
  -CWR: (0.....) CWR not significant
  -ECE: (.0.....) ECN-Echo not significant
  -Urgent: (.0.....) Not Urgent Data
  -Ack: (...1...) Acknowledgement field significant
  -Push: (...0...) No Push Function
  -Reset: (...0..0) No Reset
  -Syn: (...0...0) Not Synchronize sequence numbers
  -Fin: (...0....0) Not End of data
  Window: 16560 (scale factor 0x2) = 66240
  Checksum: 0x58AB, Disregarded
  UrgentPointer: 0 (0x0)

```

Рис. 8.10. Подтверждение запроса на соединение ACK

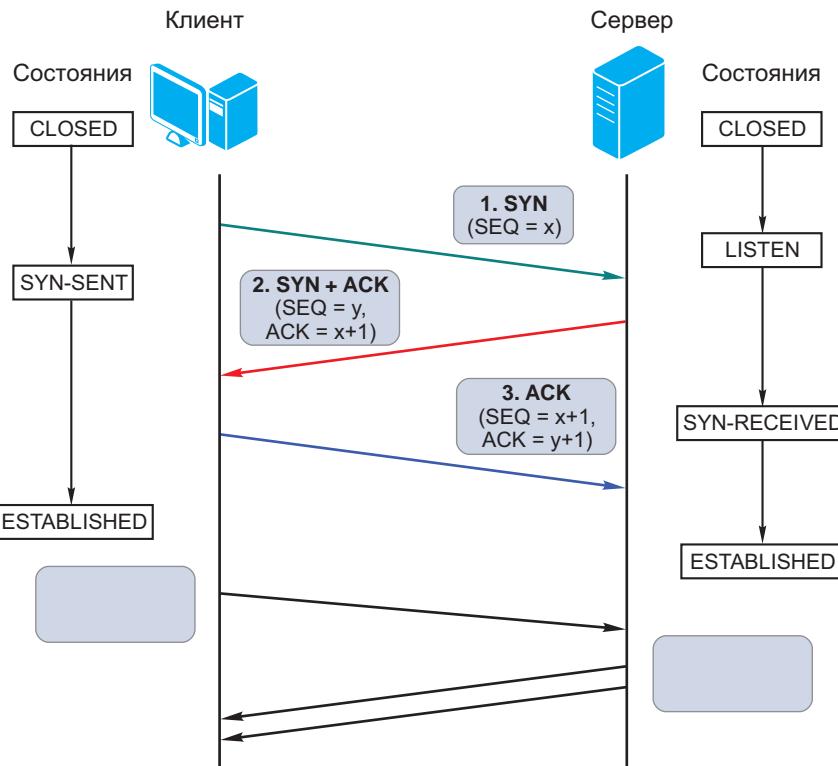


Рис. 8.11. Установка TCP-соединения

В поле *Acknowledgment Number* сегмента сервер помещает увеличенный на 1 порядковый номер клиента, тем самым подтверждая запрос на соединение (рис. 8.10) и сообщая значение порядкового номера первого байта данных, который он ожидает получить от него. В поле *Sequence Number* сервер указывает свой начальный порядковый номер, выбранный произвольно, формируя таким образом запрос на соединение.

Клиент находится в состоянии *SYN-SENT* и ожидает подтверждение ACK для отправленного им запроса и запрос на соединение SYN от сервера. Как только клиент получает сегмент SYN + ACK, он отправляет серверу сегмент с установленным битом ACK, в поле *Acknowledgment Number* которого содержится увеличенное на 1 значение начального порядкового номера сервера. После этого клиент переходит в состояние *ESTABLISHED* и может начать передачу данных.

Получив подтверждение ACK для отправленного запроса на соединение, сервер переходит в состояние *ESTABLISHED*. Теперь TCP-соединение с обеих сторон установлено, и приложения могут обмениваться данными. Процесс трехстороннего рукопожатия проиллюстрирован рис. 8.11.

TCP обрабатывает ситуацию, когда два узла одновременно пытаются установить соединение. Это нечастая ситуация, но может сложиться при определенных условиях. Порядок событий, происходящих в этом случае, показан на рис. 8.12.

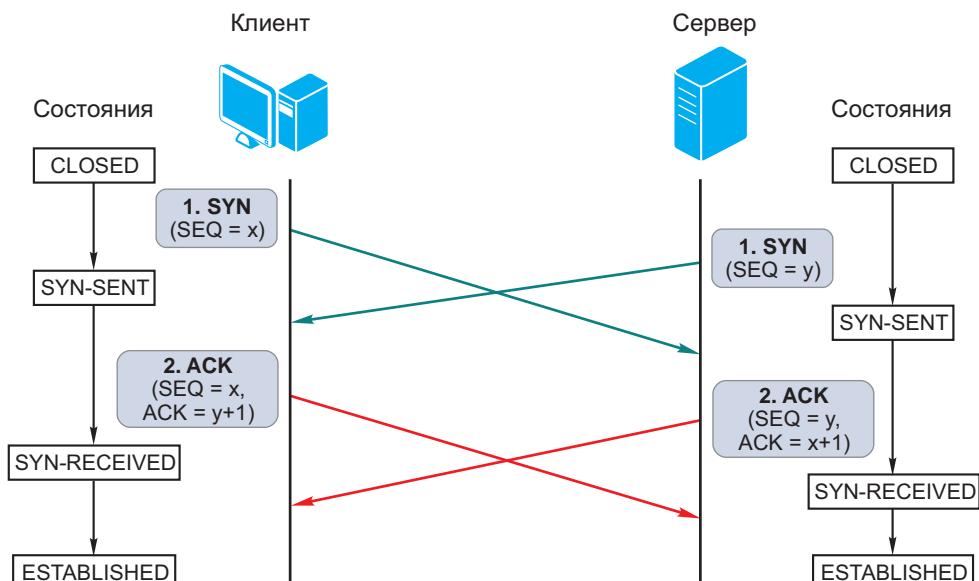


Рис. 8.12. Одновременная установка TCP-соединения

8. Протоколы транспортного уровня

Начальное значение порядкового номера, выбираемое каждым отдельным узлом для разных сессий одного и того же сокета, должно меняться, а не равняться константе. Это обеспечивает защиту от задержавшихся копий сегментов.

Помимо начального порядкового номера, в TCP-сегментах с битами SYN стороны обмениваются важными параметрами соединения, которые содержатся в поле *Options*. Одним из таких параметров является *максимальный размер сегмента* (Maximum Segment Size, MSS). Если устройства не хотят использовать размер сегмента по умолчанию, равный 536 байт, они сообщают друг другу желаемые MSS в процессе установки соединения. При этом значения MSS участников соединения могут быть разными.

Следует отметить, что схема трехстороннего рукопожатия подвержена атакам, называемым затоплением SYN-сегментами (SYN flooding attack), которые являются разновидностью атак типа «отказ в обслуживании» (Denial-of-Service).

Суть этой атаки сводится к тому, что нарушитель посыпает узлу, находящемуся в состоянии LISTEN (как правило, серверу), множество TCP-сегментов с установленным битом SYN, инициируя установление TCP-соединений, но не завершая их. Это значит, что злонамеренный отправитель может блокировать ресурсы сервера, так как он должен хранить информацию о каждом устанавливаемом соединении, создавая TCB. В результате атаки происходит переполнение таблицы соединений, и серверу не удастся установить соединение с законными пользователями.

В RFC 4987 описаны хорошо известные методы, позволяющие бороться с атаками этого типа.

8.3.4. TCP Fast Open

Как только процесс трехстороннего рукопожатия завершен, данные приложения начинают передаваться между клиентом и сервером. Клиент может начать передачу данных сразу после отправки серверу TCP-сегмента с битом ACK, а сервер должен ожидать получение этого сегмента, прежде чем передавать данные. Процесс трехстороннего рукопожатия применяется при установке любого TCP-соединения и оказывает влияние на производительность всех сетевых приложений, использующих TCP: данные приложений начнут передаваться только через время, равное круговой задержке (Round-Trip Time, RTT). *Круговая задержка* — это сумма времени, затраченного на передачу пакета, и времени, требуемого для подтверждения, что пакет был получен.

Загрузка Web-страницы часто требует подгрузки множества ресурсов с множества различных узлов. В свою очередь это может потребовать от браузера установки множества новых TCP-соединений, каждое из которых будет нести накладные расходы, связанные с трехсторонним рукопожатием. Следует отметить, что это может вызывать замедление работы браузера, особенно при подключении через медленные мобильные сети.

Для повышения производительности, требуемой современными Web-приложениями, был предложен механизм TCP Fast Open (TFO), который описан в RFC 7413. Он предлагает способ безопасной передачи данных в сегментах с битом SYN в процессе трехстороннего рукопожатия.

Стандарт TCP позволяет передавать данные в сегментах с SYN, но запрещает получателю доставлять их приложению до окончания трехстороннего рукопожатия. Это делается с целью защиты от старых или повторяющихся сегментов. TFO удаляет это ограничение и позволяет доставлять приложению данные из сегментов SYN.

Однако изменение семантики TCP накладывает новые ограничения и делает TFO неподходящим для использования определенными приложениями. Например, существует ограничение на максимальный размер данных внутри сегмента с SYN, могут отправляться только определенные HTTP-запросы и приложения должны повторно использовать соединения.

Также возникают проблемы с безопасностью. Схема трехстороннего рукопожатия подвержена атакам SYN flooding. Однако методы борьбы с ними, описанные в RFC 4987, не подходят для TFO. Для борьбы с новыми уязвимостями разработчики TFO предложили использовать генерируемые сервером cookie, но данный метод требует тщательного тестирования и доработки.

Кратко рассмотрим работу TFO. Его основным компонентом является Fast Open Cookie (cookie) — код аутентификации сообщения, генерируемый сервером (рис. 8.13). Клиент запрашивает cookie у сервера и использует его

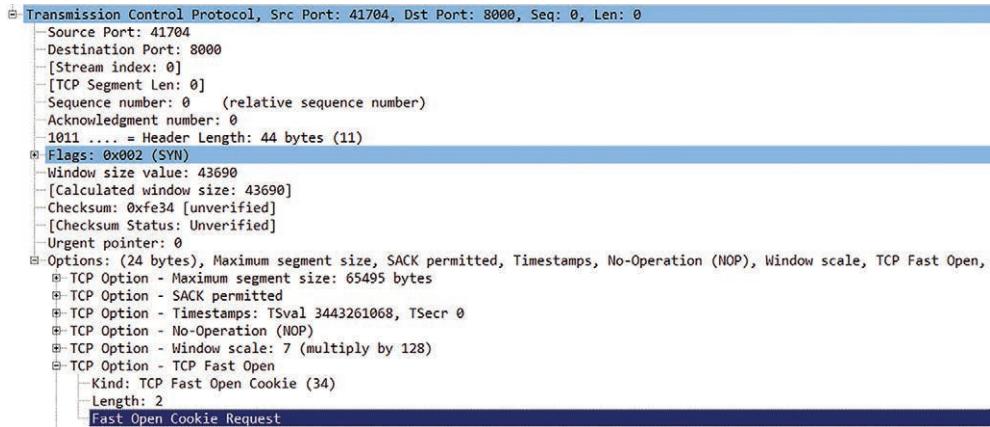


Рис. 8.13. Запрос Fast Open Cookie

в последующих запросах к тому же серверу для ускоренного обмена данными (рис. 8.14). Для запроса или отправки cookie в TCP-сегменте используется опция Fast Open.

8. Протоколы транспортного уровня

Запрос Fast Open Cookie состоит из следующих шагов (рис. 8.15):

1. Клиент отправляет сегмент с битом SYN, опция Fast Open которого содержит пустое поле cookie.
2. Сервер генерирует cookie и отправляет его в опции Fast Open сегмента с установленными битами SYN и ACK.

```
Transmission Control Protocol, Src Port: 8000, Dst Port: 41704, Seq: 0, Ack: 1, Len: 0
  Source Port: 8000
  Destination Port: 41704
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0      (relative sequence number)
  Acknowledgment number: 1    (relative ack number)
  1101 .... = Header Length: 52 bytes (13)
  Flags: 0x012 (SYN, ACK)
  Window size value: 43690
  [Calculated window size: 43690]
  Checksum: 0xfe3c [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, TCP Fast Open.
    TCP Option - Maximum segment size: 65495 bytes
    TCP Option - SACK permitted
    TCP Option - Timestamps: TSval 3443261068, TSecr 3443261068
    TCP Option - No-Operation (NOP)
    TCP Option - Window scale: 7 (multiply by 128)
    TCP Option - TCP Fast Open
      Kind: TCP Fast Open Cookie (34)
      Length: 10
      Fast Open Cookie: 7502bf93e73baf52
```

Рис. 8.14. Сегмент с cookie

3. Клиент запоминает cookie для будущего использования.

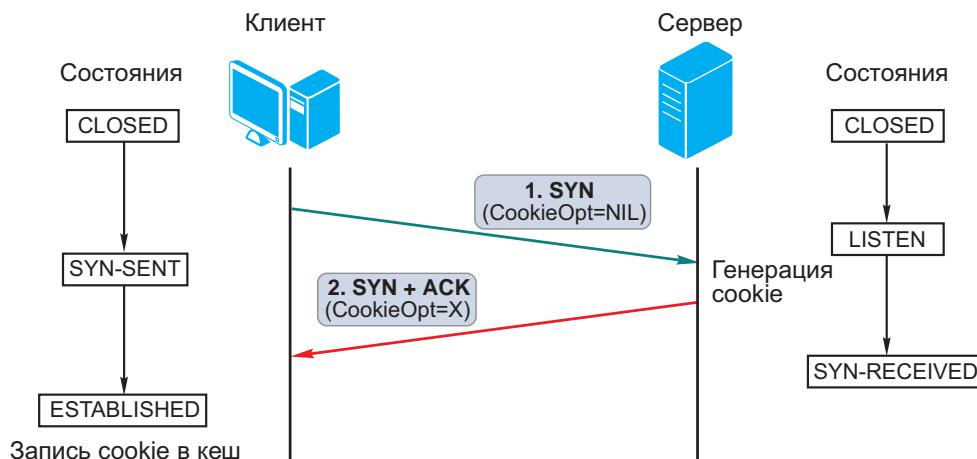


Рис. 8.15. Процесс запроса Fast Open Cookie

Выполнение TCP Fast Open:

- Клиент отправляет запрос на соединение SYN вместе с данными и cookie в опции Fast Open (рис. 8.16).

The screenshot shows a single TCP segment from a client to a server. The segment contains a SYN flag (0x0002), a cookie (7502bf93e73baf52), and some TCP options related to Fast Open. The payload contains the cookie value.

```
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 41706, Dst Port: 8000, Seq: 0, Len: 6
    Source Port: 41706
    Destination Port: 8000
    [Stream index: 1]
    [TCP Segment Len: 6]
    Sequence number: 0      (relative sequence number)
    [Next sequence number: 7  (relative sequence number)]
    Acknowledgment number: 0
    1101 .... = Header Length: 52 bytes (13)
    Flags: 0x0002 (SYN)
    Window size value: 43690
    [Calculated window size: 43690]
    Checksum: 0xfe42 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, TCP Fast Open,
        TCP Option - Maximum segment size: 65495 bytes
        TCP Option - SACK permitted
        TCP Option - Timestamps: TSval 3443270258, TSecr 0
        TCP Option - No-Operation (NOP)
        TCP Option - Window scale: 7 (multiply by 128)
        TCP Option - TCP Fast Open
            Kind: TCP Fast Open Cookie (34)
            Length: 10
            Fast Open Cookie: 7502bf93e73baf52
                [Expert Info (Note/Sequence): TCP SYN with TFO Cookie]
        TCP Option - No-Operation (NOP)
        TCP Option - No-Operation (NOP)
        [SEQ/ACK analysis]
    TCP payload (6 bytes)
        Data: 68656c6c6f21
        [Length: 6]
Data (6 bytes)
```

Рис. 8.16. Запрос на соединение SYN вместе с данными и cookie

2. Сервер проверяет cookie:

а) Если cookie верна, сервер отправляет подтверждение SYN+ACK, подтверждая и запрос на соединение, и данные. После этого сервер доставляет данные приложению;

б) В противном случае сервер отбрасывает данные и отправляет подтверждение SYN+ACK только для порядкового номера SYN.

3. Если сервер принимает данные из сегмента SYN, он может отправить в ответ свои данные, не дожидаясь окончания трехстороннего рукопожатия. Максимальный размер данных определяется контролем перегрузок TCP (RFC 5681).

4. Клиент отправляет ACK, подтверждая запрос на соединение SYN и данные сервера. Если данные клиента не были подтверждены, он повторно отправляет их в сегменте ACK.

5. Далее продолжается процесс передачи данных, как в стандартном TCP.

Согласно исследованиям, TFO позволяет сократить время загрузки Web-страниц в интервале от 10 % до 40 %. В настоящее время TFO поддерживается операционными системами Apple, Linux kernel v4.1 и выше, браузерами Google Chrome (в Linux, Chrome OS и Android) и Microsoft Edge

8. Протоколы транспортного уровня

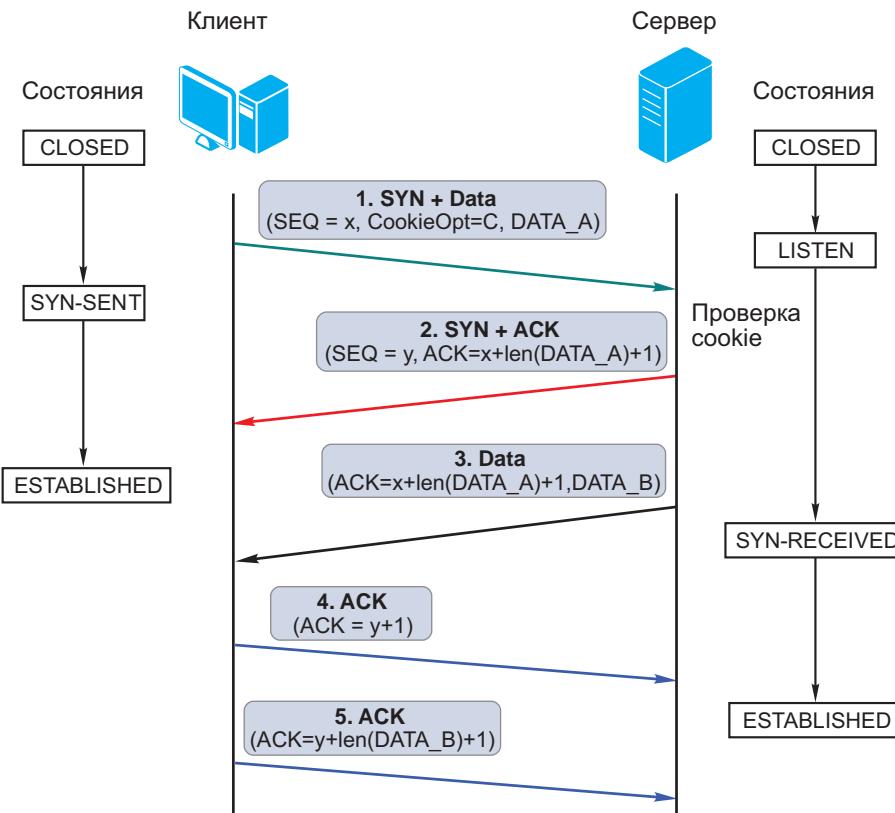


Рис. 8.17. Выполнение TCP Fast Open

(Windows 10). Для получения преимуществ от использования TFO он должен поддерживаться клиентом, сервером и приложением. В противном случае устанавливается стандартное TCP-соединение.

Активация TFO в Microsoft Edge

Функция TFO в Mozilla Firefox по умолчанию отключена. Чтобы ее включить, необходимо выполнить следующие шаги:

1. Запустить Mozilla Firefox.
 2. Перейти в скрытые настройки. Для этого в адресной строке ввести команду `about:config`.
 3. В строке Поиск ввести `network.tcp.tcp_fastopen_enable`. Изменить значение на `true`.
 4. Перезапустить браузер.
- По умолчанию в Windows 10 функция TCP Fast Open включена. Проверить настройки можно через командную строку с помощью команды `netsh interface tcp show global` (рис. 8.18).

В ОС Linux активировать механизм TFO для клиента и сервера можно с помощью команды `sudo sysctl -w net.ipv4.tcp_fastopen=3`.

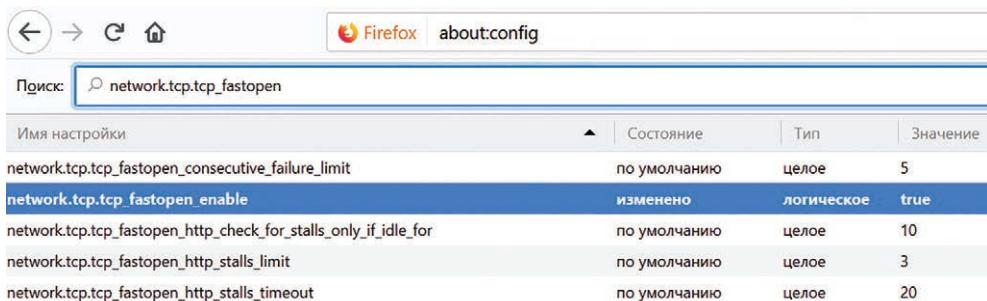


Рис. 8.18. Настройка TFO в Mozilla Firefox

8.3.5. Подтверждения и повторная передача

Каждый байт данных, отправляемый TCP, имеет порядковый номер, соответственно, его получение может быть подтверждено. Значение в поле *Номер подтверждения (Acknowledgment Number)* сегмента относится к следующему по порядку ожидаемому байту, а не к последнему полученному. В TCP используются *накопительные подтверждения (cumulative acknowledgement)* так как один номер объединяет в себе информацию обо всех полученных данных. Другими словами, подтверждение порядкового номера N говорит о том, что все байты до $N - 1$ были успешно приняты. Этот механизм позволяет обнаруживать дублирование сегментов в случае повторной передачи и потерю данных.

В протоколе TCP используются много различных таймеров. Наиболее важным из них является *таймер повторной передачи (RTO, Retransmission TimeOut)*, так как он влияет на производительность. Когда посыпается сегмент с данными, запускается таймер повторной передачи. Если подтверждение получено раньше, чем истекает время таймера, таймер останавливается. Если время таймера истечет раньше, чем прибудет подтверждение, сегмент передается еще раз, а таймер запускается снова.

Возникает вопрос, каким должен быть период времени, определенный таймером? Задержка передачи зависит от множества факторов. Предсказать, сколько потребуется времени для прохождения данных от отправителя до получателя и обратно, весьма непросто. Значение таймера RTO должно быть привязано к измерениям круговой задержки (RTT). Если выбрать время ожидания меньше фактического значения RTT, возникнут излишние повторные передачи, даже если сегменты передаются без ошибок. Если установить

8. Протоколы транспортного уровня

значение таймера больше, то из-за увеличения времени ожидания в случае потери пакета уменьшится производительность.

В TCP применяется динамический алгоритм, постоянно изменяющий значение RTO, основываясь на измерениях RTT. Для вычисления RTO отправитель использует две переменные: SRTT (smoothed round-trip time, усредненная круговая задержка) и RTTVAR (round-trip time variation, отклонение круговой задержки). Вычисление RTO описано в RFC 6298.

Для повторно переданных пакетов измерения RTT не выполняются, а используется удвоение RTO до тех пор, пока сегменты не будут передаваться с первой попытки.

8.3.6. Завершение соединения TCP

Нормальное завершение TCP-соединения использует специальную процедуру, где каждое устройство независимо от другого закрывает свое соединение. Чтобы завершить соединение, любое из устройств может отправить TCP-сегмент с установленным битом FIN. Это означает, что у него больше нет данных для передачи. Когда этот TCP-сегмент получает подтверждение ACK, передача данных от устройства завершается. Тем не менее данные могут продолжать передаваться неопределенно долго от противоположного устройства. Соединение разрывается, когда оба устройства отправили

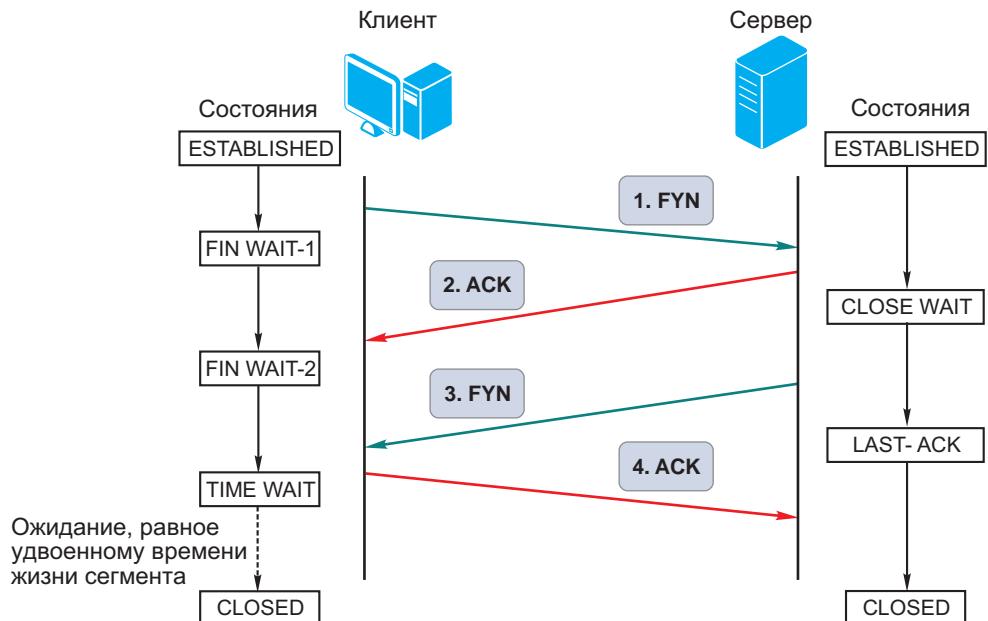


Рис. 8.19. Завершение TCP-соединения

TCP-сегмент с битом FIN и получили от противоположной стороны подтверждение ACK.

Таким образом, для нормального завершения соединения требуются четыре TCP-сегмента: по одному с битом FIN и по одному с битом ACK от каждого устройства (рис. 8.19). Первый бит ACK и второй бит FIN могут также содержаться в одном TCP-сегменте, что уменьшит количество сегментов до трех.

Прежде чем перейти в состояние *CLOSED*, устройство, последним отправившее TCP-сегмент с битом ACK, ожидает период времени, равный удвоенному максимальному времени жизни сегмента (Maximum Segment Life, MSL).

8.3.7. Механизм скользящего окна

Управление потоком (Flow control) — это механизм, который помогает предотвратить потерю данных в случае переполнения буфера принимающего устройства. Для управления потоком данных в протоколе TCP используется механизм скользящего окна (*sliding window*) (рис. 8.20). Окно определяет количество байтов, которое одна система может отправить другой без подтверждения. Для этого каждая из сторон TCP-соединения объявляет свой собственный размер приемного окна (*receive window, rwind*), т. е. размер свободного пространства в буфере принимаемых данных.

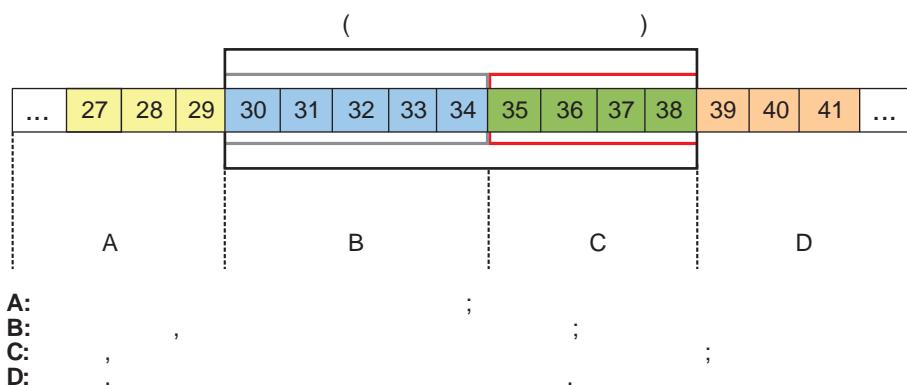


Рис. 8.20. Концепция окна, используемая в TCP

Каждая из сторон TCP-соединения должна отслеживать передаваемые и получаемые данные. Это делается путем разделения потока байтов на категории. Передаваемые байты делятся на четыре категории (рис. 8.14): переданные и подтвержденные байты; переданные, но еще не подтвержденные байты; байты, которые могут быть отправлены без подтверждения;

8. Протоколы транспортного уровня

байты, которые еще не могут быть отправлены. Получаемые байты можно разделить на три категории: полученные и подтвержденные; не полученные, но ожидаемые от отправителя; не полученные и не ожидаемые от отправителя.

Для отслеживания байтов, находящихся в каждой категории, используются специальные указатели. Они хранятся в блоках управления передачей (TCB) TCP-подсистем и обновляются при каждом обмене данными между ними. Для этого в формате TCP-сегмента предусмотрены три важных поля. Поле *Порядковый номер (Sequence Number)* указывает номер первого байта передаваемых данных. Поле *Номер подтверждения (Acknowledgment Number)* используется для подтверждения полученных данных и указывает следующий ожидаемый порядковый номер байта. Поле *Размер окна (Window)* сообщает устройству, отправляющему сегмент, количество ожидаемых для приема данных.

Первоначальный размер окна определяется сторонами при установке соединения. В процессе передачи данных размер окна изменяется. Он отражает количество свободных байтов в буфере принимаемых пакетов каждой из сторон соединения (рис. 8.21). Прежде чем передавать очередной сегмент с данными, отправитель должен получить подтверждение от приемника. Вместе с подтверждением получатель сообщает размер окна, который выражается в количестве байтов, которое он готов принять.

Посылая подтверждение, приемник подтверждает получение диапазона байтов. Отправитель, получив подтверждение, перемещает эти байты из категории «переданные, но еще не подтвержденные» в категорию «переданные и подтвержденные». Окно при этом смещается или скользит вправо на количество подтвержденных байтов, позволяя устройству передавать следующие байты потока.

Свободное пространство зависит от того, насколько быстро приложение читает данные из буфера. TCP хранит данные в буфере до тех пор, пока они не будут прочитаны приложением. Если приложение читает данные с той же скоростью, с которой их посыпает отправитель, размер окна остается примерно равным размеру буфера приемника. Таким образом, большой размер окна может повысить производительность.

Если приемный буфер полный, получатель отправляет подтверждение, в котором объявляет размер окна равным 0. Отправитель должен прекратить передачу до тех пор, пока получатель не освободит место в буфере и не увеличит размер окна.

При нулевом размере окна отправитель не может посыпать сегменты, за исключением двух случаев. Во-первых, разрешается посыпать срочные данные, например, чтобы пользователь мог уничтожить процесс, выполняющийся на удаленном устройстве.

Во-вторых, отправитель может послать 1-байтовый сегмент, прося получателя повторить информацию о размере окна и ожидаемом следующем

байте. Такой пакет называется *пробным сегментом (window probe)*. Стандарт TCP явно предусматривает эту возможность для предотвращения тупиковых ситуаций в случае потери объявления о размере окна.

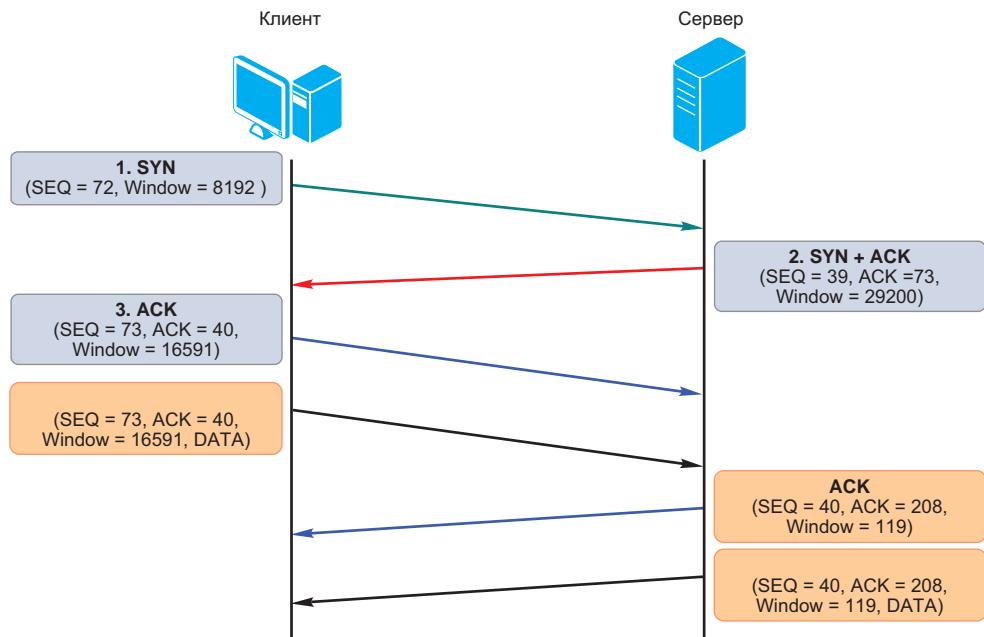


Рис. 8.21. Управление размером окна

Отправители не обязаны передавать данные сразу же, как только они приходят от приложения. Также никто не требует от получателей посыпать подтверждения как можно скорее. Эта свобода действий может использоваться для улучшения производительности.

Опция Window Scale

Поле Размер окна (Window) в заголовке сегмента TCP равно 16 битам, что ограничивает размер окна значением 2^{16} , или 65 535 байтами. Этого недостаточно, чтобы добиться оптимальной производительности, особенно в сетях с высокой пропускной способностью.

Опция TCP Window Scale (RFC 7323) (рис. 8.22) позволяет поднять максимальный размер окна до 1 Гбайта. Для этого она увеличивает количество битов для кодирования размера с 16 до 30. В опции, которая передается только в процессе трехстороннего рукопожатия в сегментах SYN, указывается коэффициент масштабирования. В дальнейшем для этого соединения изменить его нельзя. Коэффициент масштабирования совместно

8. Протоколы транспортного уровня

со значением из поля *Window* используется для определения размера окна. Максимальное значение коэффициента масштабирования равно 14, так как максимальный размер окна не может превышать 1 Гбайт ($2^{(14+16)}$).

The screenshot shows a network capture in Wireshark. A specific TCP SYN packet is selected, and its details pane is displayed. In the 'Flags' field, the value '0x002 (SYN)' is shown. Below it, the 'Options' section is expanded, revealing various TCP options. The 'TCP Option - Window scale' option is highlighted with a blue selection bar. This option has a 'Kind' of 'Window Scale (3)', a 'Length' of 3, and a 'Shift count' of 7, with a note '[Multiplier: 128]'.

```
Frame 15: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: HonHaiPr_28:10:b2 (7c:e9:d3:28:10:b2), Dst: D-Link_83:9b:5e (00:24:01:83:9b:5e)
Internet Protocol Version 4, Src: 10.255.255.150, Dst: 192.168.10.1
Transmission Control Protocol, Src Port: 53924, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 53924
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0 (relative sequence number)
    Acknowledgment number: 0
    1010 .... = Header Length: 40 bytes (10)
    Flags: 0x002 (SYN)
    Window size value: 29200
    [Calculated window size: 29200]
    Checksum: 0xab67 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
        TCP Option - Maximum segment size: 1460 bytes
        TCP Option - SACK permitted
        TCP Option - Timestamps: TSval 1144062, TSecr 0
        TCP Option - No-Operation (NOP)
        TCP Option - Window scale: 7 (multiply by 128)
            Kind: Window Scale (3)
            Length: 3
            Shift count: 7
            [Multiplier: 128]
```

Рис. 8.22. Опция Window Scale

В настоящее время функция TCP Window Scale поддерживается всеми основными операционными системами по умолчанию. В ОС Linux проверить и изменить настройки Window Scale можно с помощью команд:

```
sysctl net.ipv4.tcp_window_scaling
sysctl -w net.ipv4.tcp_window_scaling=1
```

8.3.8. Контроль и предотвращение перегрузки в TCP

Когда в какую-либо сеть поступает больше данных, чем она способна обработать, в ней образуются заторы. Наиболее часто перегрузка сети возникает в местах соединения сетей с разной полосой пропускания (рис. 8.23).

В случае возникновения перегрузки сети пакеты начинают буферизоваться и распределяться по очередям. Традиционной политикой обработки пакетов в случае переполнения всех очередей является их отбрасывание, которое продолжается до тех пор, пока длина очередей не уменьшится за счет передачи находящихся в них пакетов. Такой алгоритм управления длинной очередей получил название «отбрасывание хвоста» (*Tail-Drop*). Отбрасывание пакета будет служить сигналом о перегрузке сети источнику



Рис. 8.23. Возникновение перегрузки в сети

TCP-соединения, так как он не получит подтверждения о доставке от приемника TCP-соединения.

В стандартную реализацию TCP (RFC 5681) были добавлены четыре алгоритма контроля перегрузки:

- Slow start;
- Congestion avoidance;
- Fast retransmit;
- Fast recovery.

Контроль перегрузки позволяет поддерживать такую скорость передачи, которая не приводит к заторам в сети и потерям производительности.

Алгоритмы Slow start и Congestion avoidance

Алгоритмы медленного старта (Slow start) и предотвращения перегрузки (Congestion avoidance) являются независимыми алгоритмами, разработанными для разных целей, но на практике используются совместно. Они позволяют отправителю контролировать количество данных, отправляемых в сеть.

Для выполнения этих алгоритмов в TCP добавляются две переменные:

• *Окно перегрузки (congestion window, cwnd)* — определяет количество данных, которое отправитель может передать в сеть до получения подтверждения (ACK);

• *Порог медленного старта (slow start threshold, ssthresh)* — используется для выбора алгоритма контроля передачи данных: slow start или congestion avoidance.

Начало передачи в сеть с неизвестными условиями требует от TCP исследования ее пропускной способности, чтобы избежать отправки слишком большого количества пакетов. Измерить пропускную способность между двумя устройствами можно путем обмена данными. Для этих целей в начале передачи данных или после обнаружения их потери используется алгоритм *медленного старта*.

При установке соединения отправитель задает значение *окна перегрузки (cwnd)*. Это значение переменное. Оно не объявляется и не обменивается сторонами TCP-соединения. Возникает вопрос, как стороны определят оптимальное значение cwnd?

Отправитель начинает с маленького окна размером не более десяти сегментов (изначально стартовое значение cwnd было размером 1 сегмент;

8. Протоколы транспортного уровня

в RFC 5681 оно увеличилось до 4 сегментов, в RFC 6928 — до 10 сегментов). Максимальное количество байтов данных, которое отправитель может передать в сеть, ограничено минимальным из значений окон cwnd и rwnd. Размеры обоих окон могут изменяться в течение сессии.

Отправитель передает в сеть заданное стартовым окном перегрузки количество сегментов и ожидает их подтверждения. Получение сегментов подтверждается через время, равное круговой задержке (RTT). Для каждого полученного подтверждения отправитель увеличивает окно перегрузки на длину одного сегмента (в байтах) и может отправить два новых сегмента вместо одного. Таким образом, через интервал времени, равный круговой задержке, размер окна перегрузки увеличивается вдвое, т. е. наблюдается его экспоненциальный рост.

Так как алгоритм медленного старта начинает работу при установке TCP-соединения, стороны не могут сразу начать передачу на максимально возможной скорости. Передача начинается с небольшой скорости, которая увеличивается по мере получения подтверждений о приеме сегментов. При загрузке больших файлов или просмотре видео, т. е. при выполнении задач, требующих передачи большого потока данных, медленный старт не влияет на производительность. Максимальный размер окна будет достигнут через несколько сотен миллисекунд, и стороны продолжат передачу на скорости, близкой к максимальной. Потери производительности, вносимые медленным стартом, на фоне длительного времени жизни соединения будут незаметны.

Однако для соединений с небольшим жизненным циклом передача данных может закончиться раньше, чем окно достигнет максимального размера. В результате медленный старт ограничивает производительность приложений при передаче небольшого объема данных.

В TCP предусмотрен механизм *перезапуска медленного старта*, если соединение продолжительное время не использовалось. Предполагается, что за время простоя сетевые условия могли измениться. Поэтому, если за время, большее одного интервала RTO, не был получен ни один сегмент, значение окна перегрузки уменьшается. Этот механизм влияет на производительность соединений с большим временем жизни, которые временно «простаивают» из-за бездействия пользователя. Для ее повышения на сервере можно отключить механизм перезапуска медленного старта. В ОС Linux проверить состояние и отключить его можно с помощью команд:

```
sysctl net.ipv4.tcp_slow_start_after_idle  
sysctl -w net.ipv4.tcp_slow_start_after_idle=0
```

Поскольку алгоритм медленного старта приводит к экспоненциальному росту отправляемых сегментов, в какой-то момент в сеть будет отправлено слишком много данных, что приведет к их потере в результате переполнения буферов приемных устройств.

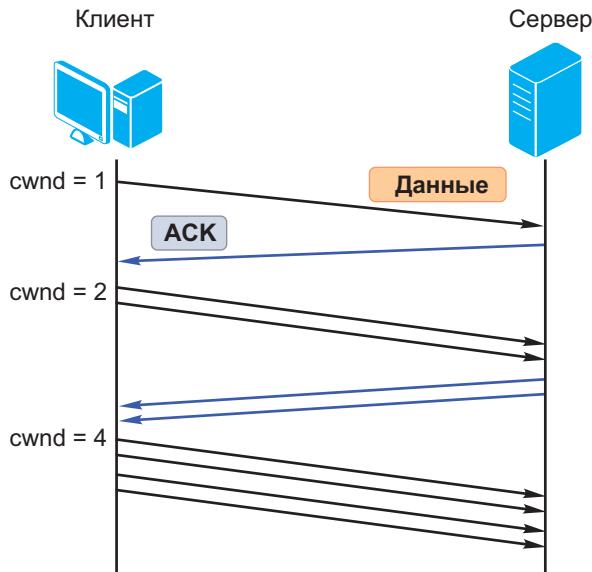


Рис. 8.24. Принцип работы алгоритма медленного старта

Чтобы контролировать медленный старт, отправитель должен хранить в памяти пороговое значение для каждого соединения. Оно называется **порогом медленного старта (slow start threshold, ssthresh)**. Изначально устанавливается его произвольное высокое значение, не превышающее размер окна получателя (rwnd), чтобы оно никак не ограничивало возможности соединения. Выполняя алгоритм медленного старта (рис. 8.24) TCP увеличивает окно перегрузки, пока оно не превысит пороговое значение ssthresh или пока не будет обнаружена перегрузка.

TCP разработан так, чтобы использовать потери сегментов в качестве сигнала о перегрузке в сети. Если от получателя не пришло подтверждение, значит, на каком-то сетевом устройстве произошло переполнение очередей, и оно отбрасывает сегменты. Индикатором перегрузки может служить таймер RTO. При обнаружении потери сегмента (после истечения RTO) порог медленного старта устанавливается следующим образом:

$$ssthresh = \max(FlightSize/2, 2*SMSS),$$

где FlightSize — количество переданных, но еще не подтвержденных байтов данных, SMSS (Sender Maximum Segment Size) — сегмент, содержащий максимальное количество байтов, которое может передать отправитель.

Кроме того, размер окна cwnd уменьшается до 1 полноразмерного сегмента (SMSS). Таким образом, при обнаружении перегрузки снижается скорость передачи. После этого алгоритм медленного старта начинает выполняться снова.

Когда размер окна cwnd достигает порогового значения, начинает работу алгоритм *предотвращения перегрузки*. Окно перегрузки будет увеличиваться на 1 сегмент SMSS через промежутки времени, равные круговой задержке. Как и в случае медленного старта, увеличение происходит по мере получения подтверждений о доставке. Однако размер окна будет расти линейно, а не экспоненциально. Линейное увеличение размера окна будет продолжаться до очередной потери сегмента. После этого процесс повторится снова.

Таким образом, алгоритм медленного старта позволяет оценить пропускную способность сети. Приблизившись к пороговому значению, выполняется переход в режим предотвращения перегрузки, в процессе которого осуществляется плавное повышение скорости передачи. Данные алгоритмы используются в версии алгоритма управления и контроля перегрузки, называемой TCP Tahoe.

Алгоритмы Fast Retransmit и Fast Recovery

Когда TCP-приемник получает сегмент, прибывший не в порядке отправления, он немедленно отправляет **повторное подтверждение (duplicate ACK)**. В нем сообщается о приеме последнего сегмента, пришедшего в порядке отправления. Повторные подтверждения имеют одинаковые номера и отсылаются до тех пор, пока не поступит копия пропущенного сегмента. После этого TCP перейдет в режим отсылки накопительных подтверждений (ACK).

Когда отправитель получает повторное подтверждение, он считает, что произошла какая-то сетевая проблема. Поскольку он не знает ее причину (потеря сегмента или его прибытие вне очереди), для принятия решения отправитель ожидает получения нескольких *duplicate ACK*.

После получения трех повторных подтверждений, что сигнализирует о потере сегмента, начинает работать алгоритм **быстрого повтора (Fast Retransmission)** (рис. 8.25). Он позволяет сразу выполнить повторную передачу отсутствующего сегмента, не дожидаясь, пока сработает таймер RTO. По номеру повторного подтверждения можно установить, какой именно сегмент потерян. Им является следующий по порядку сегмент.

После того как алгоритмом Fast Retransmission был отправлен отсутствующий сегмент, выполняется **алгоритм быстрого восстановления (Fast Recovery)**. Он позволяет не уменьшать размер окна перегрузки до одного сегмента после обнаружения потери данных, т. е. не выполнять медленный старт. До тех пор пока не будет получено подтверждение о получении повторно отправленного сегмента, будет выполняться алгоритм предотвращения перегрузки.

Совместная работа алгоритмов Fast Retransmission и Fast Recovery выполняется следующим образом.

1. При получении последовательно трех повторных подтверждений порог медленного старта устанавливается равным значению $\max(\text{FlightSize}/2, 2 \cdot \text{SMSS})$. Отсутствующий сегмент повторно передается, и окно перегрузки становится равным значению ssthresh плюс три сегмента SMSS.

2. Каждый раз при получении дополнительного (после третьего) повторного подтверждения размер окна перегрузки увеличивается на один сегмент SMSS.

3. При получении накопительного подтверждения (ACK) для повторно отправленного сегмента размер окна перегрузки становится равным порогу медленного старта, установленному на шаге 1. После этого продолжит выполняться алгоритм предотвращения перегрузки.

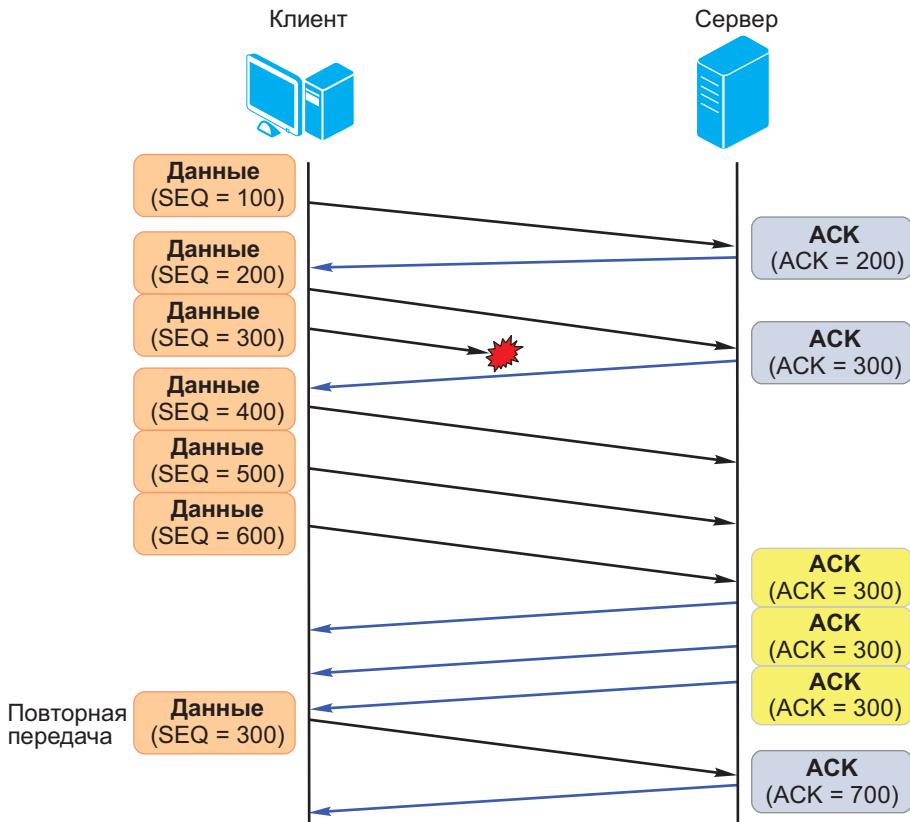
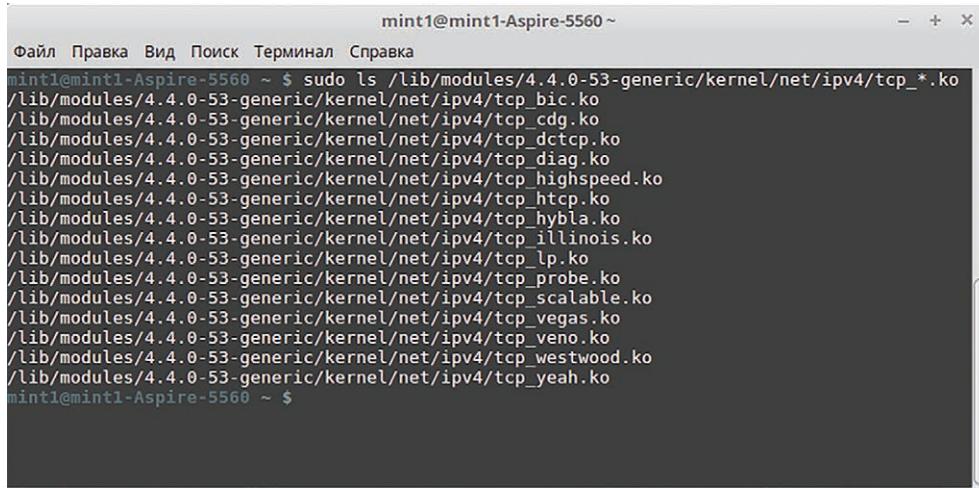


Рис. 8.25. Принцип работы алгоритма быстрого повтора

Описанные алгоритмы используются в версии алгоритма управления и контроля перегрузки, называемой TCP Reno, которая поддерживает основные принципы TCP Tahoe — медленный старт и предотвращение перегрузки.

С середины 1990-х годов стали появляться модификации TCP Reno, основанные на разных законах управления перегрузкой. К примеру, TCP Vegas, TCP New Reno, TCP BIC, TCP CUBIC (используется в Linux по умолчанию), Compound TCP (используется в Windows по умолчанию), TCP Westwood, TCP Hybla, TCP Illinois, TCP Veno, BBR (Bottleneck Bandwidth and RTT).

8. Протоколы транспортного уровня



```
mint1@mint1-Aspire-5560 ~ $ sudo ls /lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_* .ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_bic.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_cdg.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_dtcpc.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_diag.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_highspeed.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_htcp.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_hybla.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_illinois.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_lp.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_probe.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_scalable.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_vegas.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_veno.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_westwood.ko
/lib/modules/4.4.0-53-generic/kernel/net/ipv4/tcp_yeah.ko
mint1@mint1-Aspire-5560 ~ $
```

Рис. 8.26. Алгоритмы управления и контроля перегрузки TCP, поддерживаемые Linux

В ОС Linux можно изменить алгоритм, используемый по умолчанию с помощью команды `sysctl -w net.ipv4.tcp_congestion_control` (рис. 8.26).

8.3.9. Явное уведомление о перегрузке (ECN)

Как было описано выше, TCP разработан так, чтобы использовать потери сегментов в качестве сигнала о перегрузке в сети. TCP-источник определяет, что пакеты были отброшены после получения трех повторных подтверждений (ACK) или истечения времени таймера повторной передачи. В результате он уменьшает размер окна перегрузки, что снижает скорость передачи.

Отбрасывание пакетов в качестве метода уведомления о перегрузке имеет ряд недостатков. Например, оно негативно влияет на работу приложений, чувствительных к задержкам и потерям пакетов. В результате повторной передачи пакетов увеличиваются задержки.

Вместо отбрасывания пакетов разработчики TCP/IP предложили механизм, позволяющий конечным точкам соединения сообщать о перегрузке. Этот механизм получил название **Explicit Congestion Notification (ECN)**.

ECN является расширением протоколов IP и TCP (рис. 8.27). Он описан в RFC 3168. На сетевом уровне (протокол IP) узел-источник должен уметь уведомлять о своей поддержке ECN, а маршрутизирующее устройство должно уметь сообщать о перегрузке при пересылке пакета. На транспортном уровне (протокол TCP) стороны соединения должны сообщать друг другу, что они поддерживают ECN. Узел-приемник должен информировать узел-источник, что получил от маршрутизирующего устройства уведомление о перегрузке. Узел-источник должен подтверждать получение уведомления о перегрузке и снижать скорость передачи.

Для работы ECN его должны поддерживать все маршрутизаторы и коммутаторы L3 на пути между конечными точками TCP-соединения. Возможность использования ECN согласовывается конечными точками в процессе установления TCP-соединения.

RFC 3168 добавляет в заголовок IP поле ECN (рис. 8.27), которое использует два неиспользуемых бита поля DS (Differentiated Services). Напомним, что поле ToS в RFC 2474 было заменено на поле DS. Первые 6 бит поля DS называются Differentiated Services Codepoint (DSCP), два последних не используются.

Поле ECN может принимать следующие значения:

- «**00**» — узел не поддерживает ECN (код Not-ECT);
- «**01**» и «**10**» — узел поддерживает ECN (код ECT);
- «**11**» — маршрутизирующее устройство испытывает перегрузку (код CE).

В заголовок TCP-сегмента RFC 3168 добавляет два управляющих бита (флага) (см. рис. 8.27):

- **ECN-Echo (ECE)**: бит ECE-эхо используется для уведомления отправителя о перегрузке;
- **Congestion Window Reduced (CWR)**: бит используется отправителем для уведомления получателя об уменьшении окна перегрузки после получения ECN-Echo.

Когда узлы устанавливают TCP-соединение, то они обмениваются сегментами с установленными битами SYN, SYN+ACK и ACK. В случае если

The screenshot shows the detailed structure of an IP and TCP header. The IP header includes fields like Version (4 bytes), Header Length (4 bytes), Differentiated Services Field (DS field, 2 bytes), and ECN (2 bytes). The TCP header includes fields like Flags (4 bytes), CWR (1 byte), ECE (1 byte), and other standard TCP flags. Red boxes highlight the DS field in the IP header and the CWR and ECE fields in the TCP header.

```
Ipv4: Src = 192.168.1.3, Dest = 173.194.44.4, Next Protocol = TCP, Packet ID = 31934, Total IP Length: 52 (0x34)
Versions: IPv4, Internet Protocol; Header Length = 20
DifferentiatedServicesField: DSCP: 0, ECN: 0
  DSCP: (000000..) Differentiated services codepoint 0
  ECT: (.....0.) ECN-Capable Transport not set
  CE: (.....0) ECN-CE not set
TotalLength: 52 (0x34)
Identification: 31934 (0x7CBE)
FragmentFlags: 16384 (0x4000)
TimeToLive: 128 (0x80)
NextProtocol: TCP, 6(0x6)
Checksum: 58003 (0xE293)
SourceAddress: 192.168.1.3
DestinationAddress: 173.194.44.4
Tcp: Flags=.....S., SrcPort=1696, DstPort=HTTPS(443), PayloadLen=0, Seq=2997456424, Ack=0, Win=8192
  SrcPort: 1696
  DstPort: HTTPS(443)
  SequenceNumber: 2997456424 (0xB2A98E28)
  AcknowledgementNumber: 0 (0x0)
  DataOffset: 128 (0x80)
  Flags: .....S.
    CWR: (0.....) CWR not significant
    ECE: (0.....) ECN-Echo not significant
    Urgent: (..0....) Not Urgent Data
    Ack: (...0...) Acknowledgement field not significant
    Push: (...0...) No Push Function
    Reset: (.....0..) No Reset
    Syn: (.....1.) Synchronize sequence numbers
    Fin: (.....0) Not End of data
  Window: 8192 ( Negotiating scale factor 0x2 ) = 8192
```

Рис. 8.27. Поля ECN в протоколах IP и TCP

8. Протоколы транспортного уровня

узел поддерживает ECN, в сегменте с битом SYN также должны быть установлены биты ECE и CWR. В сегменте с битами SYN+ACK устанавливается только бит ECE. Бит CWR не устанавливается.

Если узел не хочет использовать ECN, в сегменте с битом SYN не устанавливаются биты ECE и CWR. В сегменте с битами SYN+ACK также не устанавливаются биты ECE и CWR.

После успешного установления TCP-соединения начинается передача данных. Узел с поддержкой ECN отправляет сегменты в IP-пакетах, заголовки которых содержат в поле ECN значения 01 или 10 (код ECT). Если на пути между отправителем и получателем находится маршрутизатор или коммутатор L3 с поддержкой ECN, испытывающий перегрузку, он не отбрасывает пакет, а изменяет значение поля ECN в IP-заголовке на 11 (код CE) (рис. 8.28). Когда сетевой уровень получателя принимает такой пакет, он сигнализирует протоколу TCP, что произошла перегрузка. TCP-приемник должен информировать об этом TCP-источнике. Он отправляет сегмент ACK с установленным битом ECE в заголовке TCP (рис. 8.29).

Получив сегмент с установленным битом ECE, TCP-источник уменьшает размер окна cwnd, выполняя алгоритмы медленного старта и предотвращения перегрузки. В TCP-заголовке следующего отправляемого сегмента он устанавливает бит CWR, подтверждая получение уведомления и сообщая об уменьшении размера окна.

ECN поддерживается современными операционными системами. По умолчанию он отключен.

Для включения ECN в ОС Windows используется команда:

```
netsh interface tcp set global ecncapability=enabled
```

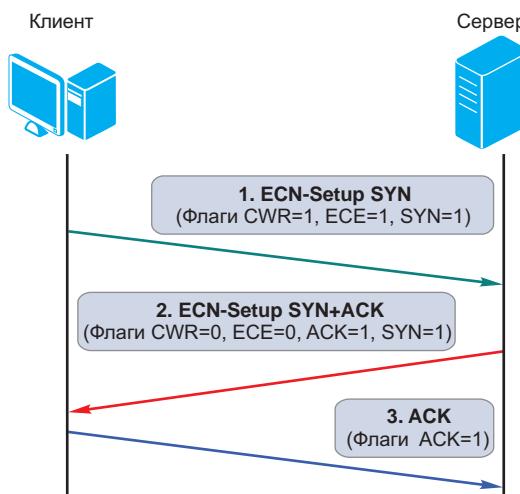


Рис. 8.28. Переговоры о поддержке ECN

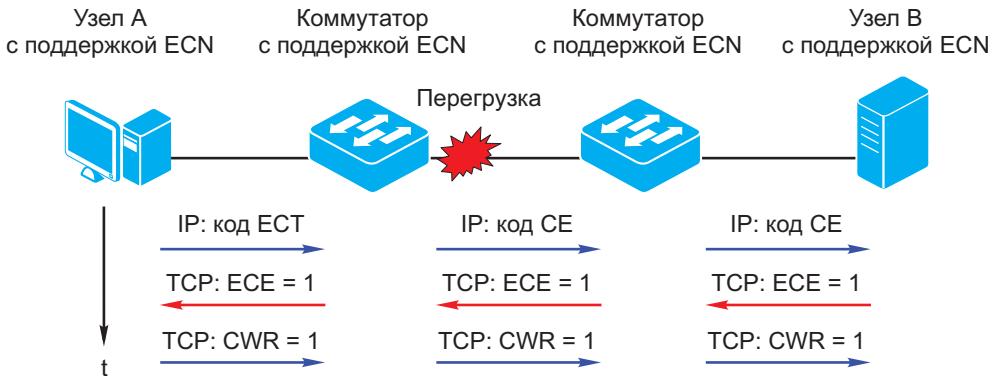


Рис. 8.29. Явное уведомление о перегрузке

Для активации ECN в ОС Linux используется команда:

```
sysctl net.ipv4.tcp_ecn=1
```

На коммутаторах механизм L3 ECN используется совместно с алгоритмом WRED (Weighted Random Early Detection). Поэтому для работы ECN должен быть настроен профиль WRED. Алгоритм WRED определяет, требуется или нет отбрасывать пакет на основе размера очереди. Если размер очереди превышает некоторое предопределенное значение и принятый пакет должен быть отброшен, коммутатор проверяет значение поля ECN в заголовке IP. Если там содержится код ECT, пакет не отбрасывается. Коммутатор изменяет код ECT на CE и пересыпает его дальше. Если в IP-заголовке указан код Not-ECT, пакет отбрасывается. Если указан код CE, коммутатор пересыпает пакет без изменений.

8.3.10. Функция Virtual Server

Давайте вернемся назад и вспомним работу механизма NAT. Он преобразует адреса из частного адресного пространства сети в однозначное и зарегистрированное открытое адресное пространство IPv4. Существуют два варианта традиционного NAT, называемые базовым NAT и NAPT (Network Address Port Translation). При базовом NAT в исходящих из частной сети пакетах NAT-маршрутизатор заменяет локальный IP-адрес источника на внутренний глобальный IP-адрес из пула адресов и вносит запись в таблицу NAT, где фиксируется соответствие IP-адресов. Затем он рассчитывает новую контрольную сумму для заголовка IP, и измененный пакет IP с новым заголовком передается адресату. Адрес получателя в пакете не изменяется. Преобразование адресов выполняется один к одному.

NAPT (или Port Address Translation (PAT), или overloaded NAT) дополнительно преобразует идентификатор транспорта, такой как номера портов TCP и UDP. Механизм NAPT позволяет большому числу узлов частной сети разделять единственный глобальный адрес. Чтобы можно было различать

8. Протоколы транспортного уровня

IP-пакеты разных отправителей, устройство NAPT заменяет (возможно, не однозначный) номер порта TCP/UDP в заголовке TCP/UDP исходного пакета IP на другой, уникальный номер порта TCP/UDP, и вносит соответствующую запись в таблицу NAT. При таком преобразовании система должна заново рассчитать контрольную сумму не только заголовка IP, но и заголовка TCP/UDP. После этого она создает новые заголовки TCP/UDP и IP и передает пакет IP соответствующему адресату. NAPT может быть скомбинирован с базовым NAT таким образом, чтобы использовался пул глобальных адресов совместно с преобразованием портов.

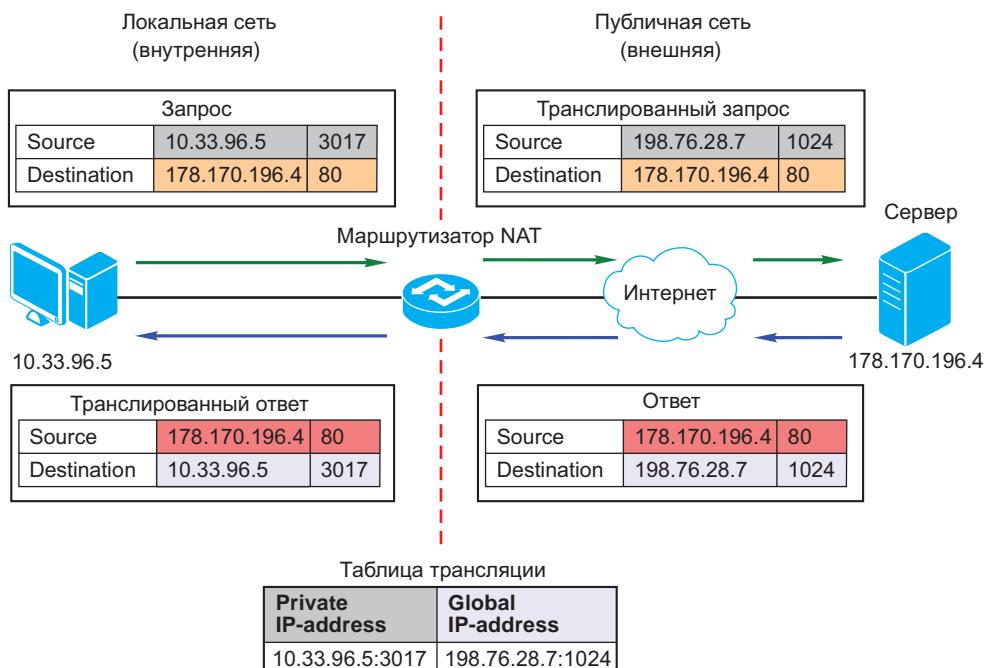


Рис. 8.30. Принцип работы NAPT

Представьте ситуацию, что во внутренней сети компании находятся общедоступные серверы. Как пользователям, находящимся за пределами внутренней сети, получить к ним доступ? На границе внешней и внутренней сети находится маршрутизатор NAT. Он будет пропускать пакеты из внешней сети во внутреннюю только в том случае, если в таблице трансляций имеются соответствующие записи. Понятно, что такие записи не могут быть созданы по причине отсутствия информации о возможных адресатах запросов и их большого количества.

Решением является использование функции *Virtual Server* (*Виртуальные серверы*). Она поддерживается маршрутизаторами D-Link серий DIR-xxx

Технологии TCP/IP в современных компьютерных сетях

и DSL-xxx. Функция позволяет создавать в таблице трансляции статические записи, обеспечивающие передачу пакетов из внешней сети на внутренние серверы (рис. 8.31). Обращение к серверам выполняется с использованием глобального IP-адреса маршрутизатора, который транслируется в их внутренние IP-адреса. Отличить один запрашиваемый сервис от другого позволяет указание номеров портов TCP/UDP. Диапазоны портов, на которые запросы для данного сервиса приходят извне и которые «прослушивает» сервер внутри сети, могут отличаться. Они настраиваются независимо друг от друга.

С целью повышения безопасности можно разрешить доступ к внутренним серверам только конкретному устройству или устройствам из определенной подсети.

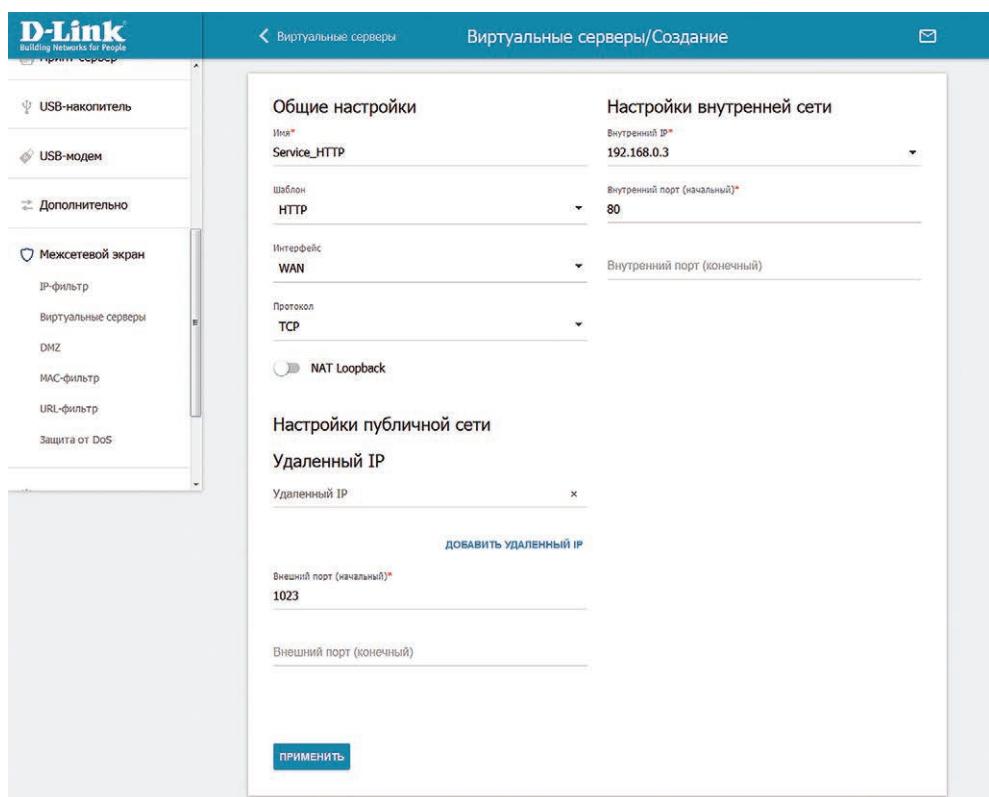


Рис. 8.31. Пример настройки функции «Виртуальные серверы» на маршрутизаторе D-Link

9. Протоколы уровня приложений

Уровень приложений (*Application layer*) находится на самом верху моделей OSI и TCP/IP. Он обеспечивает интерфейс между пользовательскими приложениями и нижележащей сетью, т. е. предоставляет приложениям возможность доступа к сервисам нижележащих уровней и определяет протоколы, с помощью которых они обмениваются данными. Существует множество протоколов уровня приложений. Рассмотрим некоторые из них, использующие при администрировании сетей.

9.1. Протокол Telnet

Telnet является старейшим среди протоколов TCP/IP. Он описан в RFC 854 и используется в качестве транспорта протокол TCP (порт 23). Telnet обеспечивает эмуляцию текстового терминала, позволяя терминалам или терминальным процессам взаимодействовать через сеть TCP/IP.

Протокол Telnet основан на трех основных идеях (рис. 9.1):

- 1) концепции сетевого виртуального терминала;
- 2) принципе согласования опций;
- 3) симметрии терминалов и процессов.

Каждый конец соединения Telnet играет роль *сетевого виртуального терминала* (*Network Virtual Terminal, NVT*). Сетевой виртуальный терминал — абстрактное устройство, которое представляет традиционный терминал. Это исключает необходимость запоминания клиентом и сервером информации о характеристиках терминалов друг друга. Таким образом, обеспечивается стандартный интерфейс при подключении к удаленному устройству. Виртуальный терминал имеет принтер, чтобы выводить на экран входящие данные, и клавиатуру, что вводить данные, которые будут переданы удаленной стороне. Для того чтобы расширить минимальные возможности виртуального терминала, Telnet предоставляет сторонам соединения средства для согласования опций (например, локальный или удаленный эхо-контроль, страничный режим, высоту и ширину экрана и т. д.). При этом и сервер, и клиент могут предлагать свои опции независимо друг от друга. Одна сторона инициирует запрос, а другая сторона может либо принять, либо отвергнуть предложение. Если запрос принимается, то опция немедленно вступает в силу.



Рис. 9.1. Сервис Telnet

После установления соединения Telnet виртуальные терминалы могут обмениваться данными двух типов: непосредственно вводимыми пользователем данными (рис. 9.2) и управляющими командами Telnet.

Транспортное соединение Telnet полнодуплексное. Однако виртуальный терминал, с точки зрения пользователя, является полуоднодуплексным устройством, работающим в буферном строковом режиме. Прежде чем отправить введенные данные, NVT помещает их в буфер. Данные пользователя рассматриваются как набор ASCII-символов и передаются без изменений.

```
Transmission Control Protocol, Src Port: 23, Dst Port: 49166, Seq: 4, Ack: 22, Len: 280
Telnet
Will Suppress Go Ahead
Will Echo
Data: \033[0m\033[1;1H\033[2J\n
Data: \r          DGS-3120-24TC Gigabit Ethernet Switch\n
Data: \r          Command Line Interface\n
Data: \r\n
Data: \r          Firmware: Build 4.19.R002\n
Data: \r          Copyright(C) 2017 D-Link Corporation. All rights reserved.\n
Data: \rUserName:
Don't Negotiate About Window Size
Don't Terminal Speed
Don't Terminal Type
Don't New Environment Option
```

Рис. 9.2. Данные пользователя в пакете Telnet

Для согласования опций и сигнализации Telnet использует специальные команды. Каждая команда может иметь длину 2 или 3 байта. Она начинается со специального символа «Interpret as Command» (IAC), за которым следует код команды. Третьим байтом в командах, служащих для согласования опций, является код опции.

Для подключения по Telnet на стороне пользователя должно быть установлено клиентское программное обеспечение, которое создает виртуальное устройство. В большинстве операционных систем поддерживается клиент Telnet. В Windows соединение Telnet может быть запущено из командной строки при условии, что клиент Telnet установлен. Также клиент Telnet реализуется эмуляторами терминала HyperTerminal, PuTTY (рис. 9.3).

Для обеспечения подключения клиентов на сервере должен быть запущен демон Telnet. Управляемые коммутаторы D-Link поддерживают сервер Telnet, и по умолчанию сервис Telnet на них активирован.

С помощью Telnet пользователь на локальном узле (клиенте) может подключиться к удаленному узлу (серверу) и выполнять на нем команды так, как если бы был подключен к нему непосредственно. Например, администратор сети может подключиться к интерфейсу командной строки удаленного маршрутизатора или коммутатора и выполнять его настройку, вводя команды устройства.

9. Протоколы уровня приложений

Несмотря на то что Telnet поддерживает аутентификацию по паролю, его применение для удаленного администрирования в настоящее время является нежелательным с точки зрения безопасности. Во-первых, используемые по умолчанию демоны Telnet имеют сетевые уязвимости. Во-вторых, протокол не предусматривает ни шифрования данных, ни проверку их подлинности, т. е. Telnet-соединение не обеспечивает безопасность передаваемых данных.

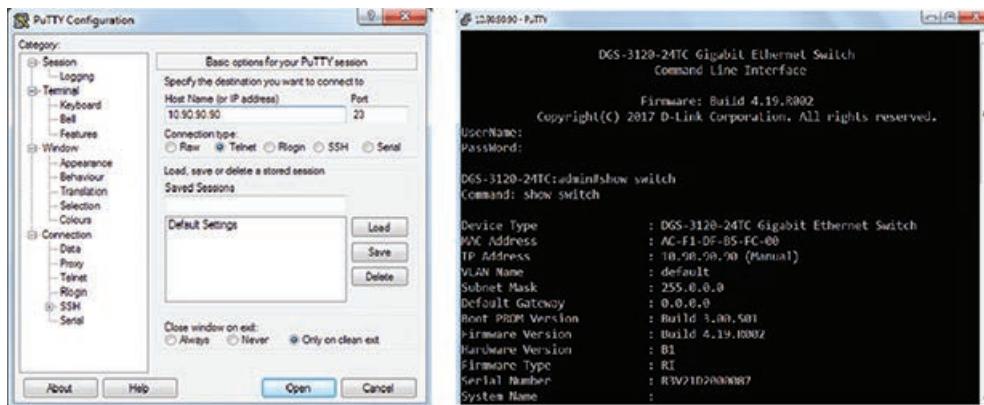


Рис. 9.3. Подключение по Telnet к коммутатору с помощью эмулятора терминала PuTTY

Однако Telnet можно использовать как средство диагностики соединения между сетевыми узлами. Поскольку он работает на уровне приложений, успешная установка соединения с удаленным устройством говорит о том, что уровни с первого по седьмой функционируют исправно.

9.2. Протокол SSH

Протокол Secure Shell (SSH) — это протокол для защиты удаленного доступа и других сетевых сервисов, выполняемых поверх незащищенной сети. Первоначальная версия протокола SSH1 была сфокусирована на обеспечение защиты удаленного подключения с целью замены Telnet, который передает логин и пароль в открытом виде, и других методов, которые не обеспечивают безопасность. Новая версия протокола SSH2 устраняет ряд уязвимостей безопасности первоначальной схемы. SSH2 задокументирован как стандарт в RFC с 4250 по 4256.

В корпоративных сетях протокол используется для следующего:

- обеспечения безопасного доступа к автоматизированным процессам;
- безопасной передачи файлов;
- безопасного удаленного ввода команд;
- управления сетевой инфраструктурой и другими важными системными компонентами.

Протокол работает на основе клиент-серверной модели. Приложения клиента и сервера SSH поддерживаются многими операционными системами. Но поскольку протокол стал широко используемым методом для удаленного администрирования и переадресации удаленных клиентов, его поддержка вышла за пределы операционных систем. Существует множество программных средств для реализации SSH. Информацию о них можно найти в Интернете.

При реализации поверх TCP/IP сервер обычно прослушивает соединения на порт 22. Этот номер порта зарегистрирован IANA для SSH. SSH состоит из трех протоколов:

- **Транспортный протокол (SSH Transport Layer Protocol)** обеспечивает аутентификацию серверов, конфиденциальность и целостность данных. Дополнительно может выполнять сжатие данных. Работает в основном с использованием соединений TCP/IP, но может быть реализован и на базе иных надежных протоколов с управлением потоком;

- **Протокол аутентификации пользователей (SSH User Authentication Protocol)** аутентифицирует пользователя на сервере. Работает поверх транспортного протокола;

- **Протокол соединений (SSH Connection Protocol)** обеспечивает мультиплексирование нескольких логических каналов в один шифрованный туннель. Работает поверх протокола аутентификации пользователей.

9.2.1. Транспортный протокол SSH

Транспортный протокол (RFC 4253) отвечает за обмен ключами и аутентификацию серверов. Аутентификация выполняется на основе обработки сервером пары ключей — открытого и закрытого. Сервер может иметь несколько хост-ключей, используя несколько различных алгоритмов асимметричного шифрования. Чтобы аутентификация стала возможной, клиент должен предварительно знать открытый хост-ключ сервера. Для этого на клиенте создается локальная база данных, в которую вручную вносится привязка каждого имени хоста к соответствующему открытому хост-ключу. Альтернативным вариантом является сертификация привязки имени хоста к ключу доверенным удостоверяющим центром (Certification Authority, CA).

Клиент SSH инициирует процесс подключения и использует открытый ключ для проверки идентичности SSH-сервера. Первоначально клиент устанавливает подключение к серверу, используя протокол TCP, UDP или SCTP, который не является частью транспортного протокола SSH. Когда это соединение установлено, клиент и сервер начинают обмен данными, который состоит из нескольких шагов.

Первым шагом является **обмен информационными строками**. Клиент отправляет пакет с идентификационной строкой в форме SSH-protoversion-softwareversion SP comments CR LF (рис. 9.4), где SP, CR и LF соответственно пробел, возврат каретки и перевод строки. Сервер в ответ отправляет свою идентификационную строку (рис. 9.5). Эти строки используются в процессе обмена ключами Диффи — Хеллмана.

5 0.025421	10.90.90.99	10.90.90.90	SSHv2	82 Client: Protocol (SSH-2.0-PuTTY_Release_0.62)
⊕	Frame 5: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0			
⊕	Ethernet II, Src: WistronI_72:a5:82 (20:6a:8a:72:a5:82), Dst: D_LinkIn_b5:fc:00 (ac:f1:df:b5:fc:00)			
⊕	Internet Protocol Version 4, Src: 10.90.90.99, Dst: 10.90.90.90			
⊕	Transmission Control Protocol, Src Port: 49167, Dst Port: 22, Seq: 1, Ack: 18, Len: 28			
⊖	SSH Protocol			
	Protocol: SSH-2.0-PuTTY_Release_0.62			

Рис. 9.4. Идентификационная строка клиента

4 0.011307	10.90.90.99	10.90.90.99	SSHv2	71 Server: Protocol (SSH-2.0-1.00.000)
⊕	Frame 4: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0			
⊕	Ethernet II, Src: D_LinkIn_b5:fc:00 (ac:f1:df:b5:fc:00), Dst: WistronI_72:a5:82 (20:6a:8a:72:a5:82)			
⊕	Internet Protocol Version 4, Src: 10.90.90.99, Dst: 10.90.90.99			
⊕	Transmission Control Protocol, Src Port: 22, Dst Port: 49167, Seq: 1, Ack: 1, Len: 17			
⊖	SSH Protocol			
	Protocol: SSH-2.0-1.00.000			

Рис. 9.5. Идентификационная строка сервера

```

SSH Protocol
  SSH Version 2 (encryption:aes256-cbc mac:hmac-sha1 compression:none)
    - Packet Length: 636
      - Padding Length: 6
        - Key Exchange
          - Message Code: Key Exchange Init (20)
            - Algorithms
              - Cookie: 4f2bea8d24bbe4337c9fb05a9eb4bd
              - kex_algorithms length: 154
              - kex_algorithms string: diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1,rsa'
              - server_host_key_algorithms length: 15
              - server_host_key_algorithms string: ssh-rsa,ssh-dss
              - encryption_algorithms_client_to_server length: 159
              - encryption_algorithms_client_to_server string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,blowfish-ctr,
              - encryption_algorithms_server_to_client length: 159
              - encryption_algorithms_server_to_client string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,blowfish-ctr,
              - mac_algorithms_client_to_server length: 31
              - mac_algorithms_client_to_server string: hmac-sha1,hmac-sha1-96,hmac-md5
              - mac_algorithms_server_to_client length: 31
              - mac_algorithms_server_to_client string: hmac-sha1,hmac-sha1-96,hmac-md5
              - compression_algorithms_client_to_server length: 9
              - compression_algorithms_client_to_server string: none,zlib
              - compression_algorithms_server_to_client length: 9
              - compression_algorithms_server_to_client string: none,zlib
              - languages_client_to_server length: 0
              - languages_client_to_server string: [Empty]
              - languages_server_to_client length: 0
              - languages_server_to_client string: [Empty]
            - First KEX Packet Follows: 0
            - Reserved: 00000000
            - Padding String: d953de80522a

```

Рис. 9.6. Список алгоритмов, поддерживаемых клиентом

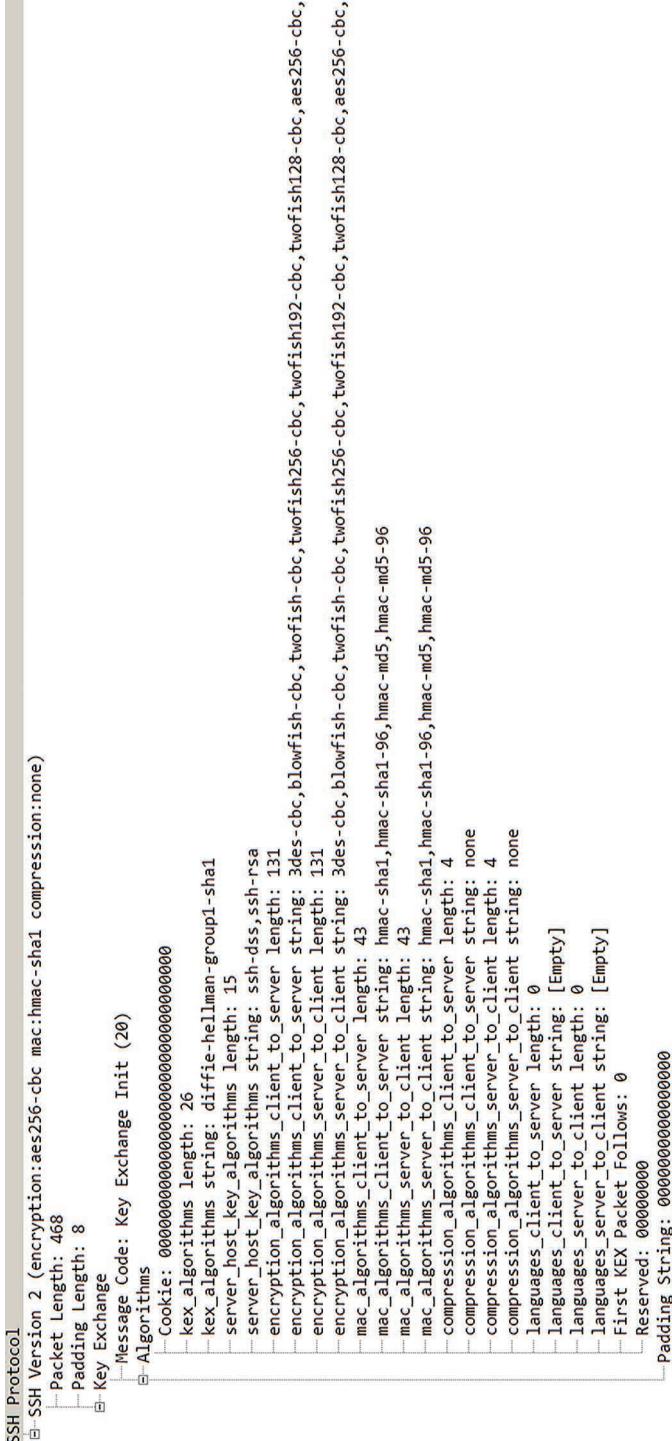


Рис. 9.7. Список алгоритмов, поддерживаемых сервером

Далее выполняется **согласование алгоритмов** шифрования, хеширования и сжатия. Стороны обмениваются списками поддерживаемых алгоритмов (рис. 9.6, рис. 9.7), которые отправляют в сообщениях `SSH_MSG_KEXINIT`. Для каждого типа алгоритма (шифрования, хеширования и сжатия) имеется отдельный список. Алгоритмы в списках расположены в порядке их предпочтения отправителем. Из каждой категории выбирается один алгоритм, поддерживаемый обеими сторонами.

После согласования алгоритмов начинается этап обмена ключами. Спецификация позволяет использовать альтернативные методы обмена ключами, но в настоящий момент в RFC 4253 определены две версии обмена ключами Диффи – Хеллмана (RFC 2409): `diffie-hellman-group1-sha1` и `diffie-hellman-group14-sha1` (рис. 9.8). Они обе требуют отправки только одного сообщения в обоих направлениях: от клиента серверу и наоборот.

Рассмотрим шаги, выполняемые в процессе обмена ключами. Сначала введем обозначение переменных:

- p — простое базовое число;
- g — генератор для подгруппы $GF(p)$;
- q — порядок подгруппы;
- V_S — идентификационная строка сервера;
- V_C — идентификационная строка клиента;
- K_S — открытый хост-ключ сервера;
- I_C — сообщение `SSH_MSG_KEXINIT` клиента, I_S — сообщение `SSH_MSG_KEXINIT` сервера, которыми они обменялись перед обменом ключами.

Значения p , g и q известны клиенту и серверу в результате переговоров по выбору алгоритмов. Хеш-функция `hash()` также определяется в процессе согласования алгоритмов.

1. Клиент генерирует произвольное число x ($1 < x < q$) и вычисляет $e = g^x \bmod p$. Он отправляет e серверу в сообщении `SSH_MSG_KEXDH_INIT`.

2. Сервер генерирует произвольное число y ($0 < y < q$) и вычисляет $f = g^y \bmod p$. Сервер получает e от клиента. Он вычисляет секретный

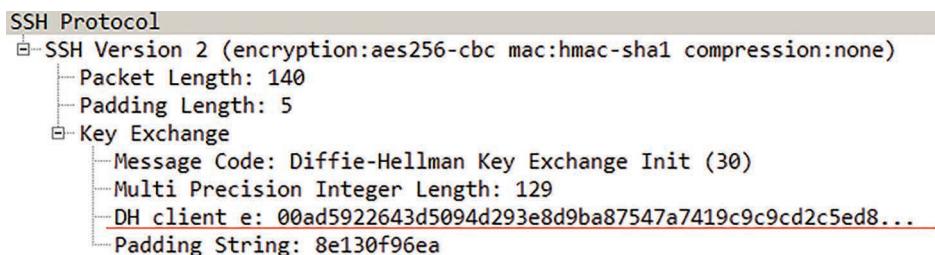


Рис. 9.8. Обмен ключами Диффи – Хеллмана: сообщение `SSH_MSG_KEXDH_INIT` клиента

9. Протоколы уровня приложений

мастер-ключ $K = ey \bmod p$, хеш $H = \text{hash}(V_C || V_S || I_C || I_S || K_S || e || f || K)$ и подпись s для H с помощью своего закрытого хост-ключа. Сервер отправляет свой открытый хост-ключ, f и подпись s клиенту в сообщении `SSH_MSG_KEXDH_REPLY`. Подписание может включать вторую операцию хеширования.

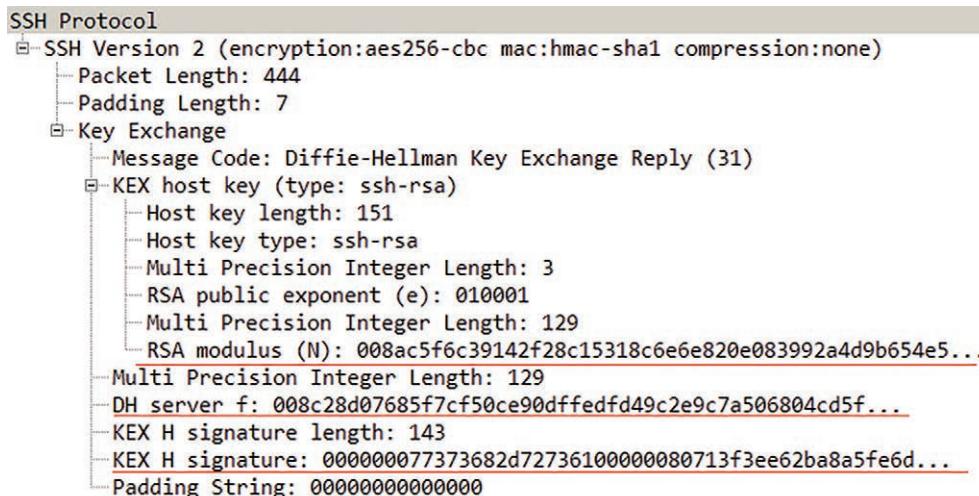


Рис. 9.9. Обмен ключами Диффи – Хеллмана: ответ сервера `SSH_MSG_KEXDH_REPLY`

3. Клиент проверяет, что открытый ключ действительно является хост-ключом сервера (используя сертификат или локальную базу данных). Клиент также может принять ключ без верификации; однако это не рекомендуется делать. Далее клиент вычисляет секретный мастер-ключ $K = fx \bmod p$, $\text{хеш } H = \text{hash}(V_C || V_S || I_C || I_S || K_S || e || f || K)$ и проверяет подпись s хеша H .

В результате выполнения этих шагов стороны знают разделяемый секретный мастер-ключ и вычисляют значение хеша. Дополнительно сервер аутентифицируется клиентом, так как он использовал свой закрытый ключ для подписи своей половины обмена. Значение хеша H служит идентификатором сессии для этого соединения. Он вычисляется один раз и не изменяется, даже если обмен ключами выполняется снова.

Об окончании обмена ключами сигнализирует сообщение `SSH_MSG_NEWKYS`, отправляемое и клиентом, и сервером. Без его отправки не начнут использоваться новые алгоритмы шифрования, хеширования и сжатия.

Финальным шагом работы протокола SSH Transport Layer является **запрос сервисов**. Клиент отправляет запрос `SSH_MSG_SERVICE_REQUEST` или

протоколу аутентификации пользователя, или протоколу соединений. Начиная с этого момента все данные передаются как нагрузка пакета протокола SSH Transport Layer и защищены шифрованием (рис. 9.10) и кодом аутентификации сообщения (Message Authentication Code, MAC). Ключи, используемые для шифрования и MAC, генерируются сторонами из разделяемого секретного мастер-ключа и значения хеша, полученных на этапе обмена ключами, и идентификатора сессии. Пакет протокола SSH Transport Layer (табл. 9.1) помещается в поле данных протокола TCP и передается по сети.

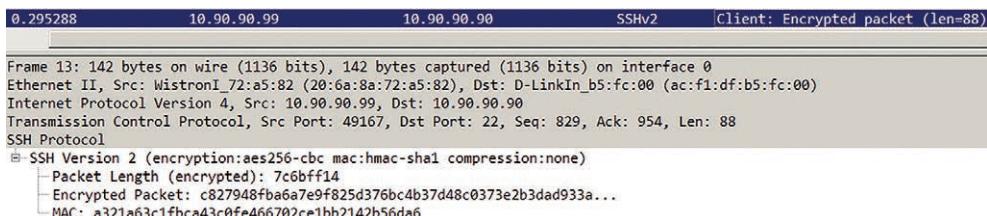


Рис. 9.10. Шифрованный пакет SSH

Таблица 9.1. Криптографические протоколы SSH Transport Layer Protocol

Название протокола	Описание
<i>Протоколы шифрования</i>	
3des-cbc	Three-key Triple Digital Encryption Standard (3DES) в режиме Cipher-Block-Chaining (CBC)
blowfish-cbc	Blowfish в режиме CBC
twofish256-cbc	Twofish в режиме CBC с ключом 256 бит
twofish192-cbc	Twofish с ключом 192 бит
twofish128-cbc	Twofish с ключом 128 бит
aes256-cbc	Advanced Encryption Standard (AES) в режиме CBC с ключом 256 бит
aes192-cbc	AES с ключом 192 бит
aes128-cbc	AES с ключом 128 бит
Serpent256-cbc	Serpent в режиме CBC с ключом 256 бит
Serpent192-cbc	Serpent с ключом 192 бит
Serpent128-cbc	Serpent с ключом 128 бит
arcfour	RC4 с ключом 128 бит
cast128-cbc	CAST-128 в режиме CBC

9. Протоколы уровня приложений

Окончание табл. 9.1

Название протокола	Описание
<i>Протоколы хеширования</i>	
hmac-sha1	HMAC-SHA1; длина хеша = длина ключа = 20
hmac-sha1-96	Первые 96 бит HMAC-SHA1; длина хеша = 12; длина ключа = 20
hmac-md5	HMAC-MD5; длина хеша = длина ключа = 16
hmac-md5-96	Первые 96 бит HMAC-MD5; длина хеша = 12; длина ключа = 16
<i>Протоколы сжатия</i>	
none	Нет сжатия
zlib	Определен в RFC 1950 и в RFC 1951

9.2.2. Протокол аутентификации пользователей SSH

Протокол аутентификации пользователей предоставляет набор методов, с помощью которых сервер аутентифицирует клиентов. Сервер может проверить подлинность клиента с помощью одного или нескольких методов:

- **Аутентификация с открытым ключом.** Выполнение этого метода зависит от выбранного алгоритма аутентификации с открытым ключом. В общем случае клиент отправляет серверу сообщение, в котором содержится его открытый ключ. Сообщение подписывается закрытым ключом клиента. Когда сервер его получает, он проверяет ключ и подпись клиента. Если ключ и подпись верны, аутентификация успешна.

- **Аутентификация по паролю.** Клиент отправляет серверу сообщение, содержащее пароль в открытом виде. Это сообщение зашифровано транспортным протоколом SSH.

- **Аутентификация узла.** Сервером SSH выполняется аутентификация клиентского устройства, а не самого клиента. Этот метод работает, когда клиент отправляет подпись, созданную с помощью закрытого ключа узла. Таким образом, все пользователи, имеющие доступ к этому устройству, будут аутентифицированы.

В протоколе используются сообщения трех типов.

1. **SSH_MSG_USERAUTH_REQUEST** — отправляется клиентом серверу с целью запроса метода аутентификации. Сообщение содержит идентификационную информацию клиента, тип запрашиваемого сервиса (обычно SSH Connection Protocol) и запрашиваемый метод аутентификации.

2. **SSH_MSG_USERAUTH_FAILURE** — отправляется сервером, если он отказал в запросе или принял запрос, но требует выполнения дополнительных методов аутентификации. Сообщение содержит поле «Частичный успех» и список методов аутентификации.

3. **SSH_MSG_USERAUTH_SUCCESS** — отправляется сервером, если он принял аутентификацию.

Обмен сообщениями включает шесть шагов.

1. Клиент отправляет серверу сообщение с запросом метода аутентификации `SSH_MSG_USERAUTH_REQUEST`.

2. Сервер проверяет, является ли клиент достоверным. Если нет, отправляется сообщение `SSH_MSG_USERAUTH_FAILURE`, поле «Частичный успех» которого содержит значение `false`. В противном случае выполняется шаг 3.

3. Сервер отправляет сообщение `SSH_MSG_USERAUTH_FAILURE`, содержащее список используемых методов аутентификации.

4. Клиент выбирает подходящий метод аутентификации и отправляет запрос `SSH_MSG_USERAUTH_REQUEST`, содержащий название метода и специфичные для метода поля. Начиная с этого момента может быть выполнено несколько обменов сообщениями, требуемых для выполнения аутентификации с помощью выбранного метода.

5. Если аутентификация успешно завершена, но сервер требует выполнения дополнительных методов аутентификации, процесс возвращается на шаг 3. Значение поля «Частичный успех» сообщения `SSH_MSG_USERAUTH_FAILURE` устанавливается равным `true`. Если аутентификация неуспешна, процесс возвращается на шаг 3, но значение поля «Частичный успех» сообщения `SSH_MSG_USERAUTH_FAILURE` устанавливается равным `false`.

6. Когда все требуемые методы аутентификации успешно завершены, сервер отправляет сообщение `SSH_MSG_USERAUTH_SUCCESS`. Протокол аутентификации пользователей заканчивает работу.

9.2.3. Протокол соединений SSH

В RFC 4254 «The Secure Shell (SSH) Connection Protocol» определено, что SSH Connection Protocol выполняется поверх транспортного протокола и протокола аутентификации пользователей. RFC 4251 «SSH Protocol Architecture» утверждает, что протокол соединений выполняется поверх протокола аутентификации пользователей. На самом деле протокол соединений выполняется поверх транспортного протокола, но предполагает, что протокол аутентификации пользователей уже используется. К моменту его работы должно существовать безопасное аутентифицированное соединение (туннель), которое он использует для мультиплексирования множества логических каналов.

Все типы соединений по SSH должны использовать отдельные каналы. Любая сторона соединения может открыть канал. Для каждого канала каждая сторона назначает уникальный номер, который не обязательно должен совпадать с номером, присвоенным каналу противоположной стороной. Для управления потоком в канале используется механизм окна. Данные не могут быть отправлены в канал до тех пор, пока от приемника не получено сообщение о его размере. Жизненный цикл канала включает три стадии: открытие канала, передача данных и закрытие канала.

Когда одна из сторон желает открыть канал, она выделяет ему локальный номер и отправляет другой стороне сообщение `SSH_MSG_CHANNEL_OPEN`. Это сообщение содержит:

- тип канала, который определяет приложение, использующее канал;
- локальный номер канала отправителя;
- первоначальный размер окна, который показывает, сколько байтов данных может быть послано отправителю этого сообщения;
- максимальный размер пакета данных, который может быть прислан отправителю.

Если удаленная сторона готова открыть канал, она высыпает подтверждающее сообщение `SSH_MSG_CHANNEL_OPEN_CONFIRMATION`. Это сообщение содержит номер канала отправителя, номер канала получателя, размеры окна и пакета для входящего трафика. В противном случае отправляется сообщение `SSH_MSG_CHANNEL_OPEN_FAILURE`, содержащее код ошибки.

После того как канал открыт, по нему в сообщениях `SSH_MSG_CHANNEL_DATA` начинают передаваться данные. Сообщения передаются в обоих направлениях. Помимо блока данных они содержат номер канала получателя.

Когда одна из сторон желает закрыть канал, она отправляет сообщение о закрытии канала `SSH_MSG_CHANNEL_CLOSE`, содержащее номер канала получателя. Канал считается полностью закрытым, когда обе стороны отправили и получили сообщения `SSH_MSG_CHANNEL_CLOSE`.

В спецификации протокола SSH Connection определены следующие типы каналов:

- **Сессия.** Сессией называется удаленное выполнение программы. Программой может быть командная строка, приложение (передачи файлов, электронной почты и т. п.),строенная подсистема. Когда канал для сессии открывается, последующие запросы используются для начала выполнения удаленной программы;

- **X11.** Этот тип канала относится к X Window System, программному обеспечению компьютера или сетевому протоколу, которые обеспечивают графический интерфейс для сетевых компьютеров. Приложения запускаются на сетевом сервере, но выводятся на экран рабочей станции;

- **TCP/IP Forwarding.** Этот тип канала используется для перенаправления портов.

Перенаправление портов

Одной из наиболее полезных функций SSH является *перенаправление портов (port forwarding)*. Она позволяет преобразовать любое незащищенное TCP-соединение в защищенное SSH-соединение. По-другому это называется SSH-туннелирование (рис. 9.11). Под портом в данном случае подразумевается номер порта приложения в контексте TCP.

Рассмотрим основную идею перенаправления портов. Клиентское приложение идентифицируется номером порта *X*, а серверное — номером порта *Y*. В какой-то момент клиентское приложение обращается к локальной TCP-подсистеме, которая запрашивает соединение с удаленным сервером для порта *Y*. Локальная TCP-подсистема устанавливает соединение с удаленной



Рис. 9.11. Подключение через SSH-туннель

TCP-подсистемой. Это соединение связывает локальный порт X с удаленным портом Y . Для защиты этого соединения SSH настраивается таким образом, чтобы транспортный протокол SSH устанавливал TCP-соединение между SSH-подсистемами клиента и сервера с номерами TCP-портов A и B соответственно. Поверх TCP-соединения создается защищенный SSH-туннель. Трафик клиентского приложения с номером порта X перенаправляется на локальную SSH-подсистему и передается через туннель удаленной SSH-подсистеме, которая доставляет данные приложению сервера с номером порта Y . Трафик в обратном направлении перенаправляется аналогично.

SSH поддерживает два типа перенаправления портов: локальное и удаленное. При *локальном перенаправлении портов* SSH конфигурируется для прослушивания выбранных портов: выбранный трафик уровня приложений перехватывается и перенаправляется в SSH-туннель. На другой стороне SSH-сервер отправляет входящий трафик на порт назначения, указанный клиентским приложением. Для пояснения приведем следующий пример. Предположим, что на компьютере имеется почтовый клиент и что вы получаете письма с почтового сервера по протоколу POP (Post Office Protocol), порт 110. Защитить трафик, передаваемый между почтовым клиентом и сервером, можно следующим образом.

1. SSH-клиент устанавливает соединение с удаленным сервером.
2. Выбираем неиспользуемый локальный номер порта, например 5432, с которого трафик будет перенаправляться на порт 110 сервера. Настраиваем SSH на прием трафика с этого порта.

9. Протоколы уровня приложений

3. SSH-клиент информирует SSH-сервер о необходимости создать соединение с почтовым сервером, порт 110.

4. SSH-клиент принимает все данные, отправляемые на локальный порт 5432, и посыпает их серверу через шифрованный туннель. SSH-сервер расшифровывает входящие данные и передает их открытым текстом на порт 110.

5. При передаче данных в обратном направлении SSH-сервер принимает все данные, отправленные на порт 110, и посыпает их через шифрованный туннель клиенту. SSH-клиент расшифровывает входящие данные и передает их процессу, подключенному к порту 5432.

При *удаленном перенаправлении портов* SSH-клиент ведет себя как сервер. SSH-клиент получает трафик на определенный порт назначения, далее изменяет его на правильный и отправляет данные клиентскому приложению. Приведем пример использования удаленного перенаправления. Допустим, что с домашнего компьютера необходимо получить доступ к серверу компании. По какой-то причине доступ через VPN недоступен. Сервер компании находится за межсетевым экраном, который не примет SSH-запрос с домашнего компьютера. Значит, на работе, используя удаленное перенаправление портов, надо установить SSH-туннель к нему. Этот процесс включает три шага:

1. Устанавливаем с рабочего компьютера SSH-соединение с домашним. Межсетевой экран пропустит это исходящее соединение.

2. Настраиваем SSH-сервер на прослушивание локального порта, пусть 2222, и доставляем данные по SSH-туннелю, например на удаленный порт 3333.

3. На домашнем компьютере SSH конфигурируется на прием трафика, поступающего на порт 3333.

В результате все соединения на порт 3333 домашнего компьютера будут туннелированы через корпоративный межсетевой экран на порт 2222 компьютера на работе. Таким образом, будет выполняться удаленное подключение к рабочему серверу.

9.3. Протоколы SSL/TLS

Протокол Secure Sockets Layer (SSL) — криптографический протокол, который обеспечивает безопасность связи. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. Протокол был разработан компанией Netscape Communications в середине 1990-х годов для выполнения безопасных финансовых транзакций через Интернет при использовании браузера Netscape Navigator. Версия SSL 1.0 никогда не была обнародована. Версия SSL 2.0, выпущенная в феврале 1995 года, была быстро заменена версией SSL 3.0 из-за обнаружения в ней ряда уязвимостей. Так как протокол SSL был собственноностью Netscape, IETF сформировала рабочую группу с целью его стандартизации. Протокол SSL переименовали в Transport Layer Security (TLS).

В январе 1999 года была опубликована первая версия TLS 1.0 (RFC 2246). Она обратно совместима с SSL 3.0, но технически TLS 1.0 и SSL 3.0 различаются, так как описывают разные версии протокола. TLS 1.0 можно рассматривать как SSL 3.1. IETF продолжил работу с протоколом TLS и расширил его возможности. Версия TLS 1.1 (RFC 4346) была опубликована в апреле 2006 года. Версия TLS 1.2 (RFC 5246) была опубликована в августе 2008 года. Версия TLS 1.3 была опубликована в августе 2018 года (RFC 8446). SSL и TLS 1.2 отличаются друг от друга лишь в деталях. Протокол TLS 1.3 сильно отличается от предшественников и напрямую с ними не совместим.

Помимо того, в RFC 3546 определено несколько расширений для случаев, когда TLS используются в системах с ограниченной пропускной способностью, таких как беспроводные сети; в RFC 6066 определен ряд расширений TLS, внесенных в протокол Handshake; RFC 7925 описывает TLS (и DTLS) при его использовании в IoT (Internet of Things, Интернете вещей).

Несмотря на то что протокол SSL 3.0 широко используется, он никогда официально не публиковался IETF. В августе 2011 года был выпущен RFC 6110, имеющий исторический статус и описывающий SSL 3.0. В июне 2015 года был опубликован RFC 7568, который признал протокол SSL 3.0 не безопасным и не рекомендованным к использованию.

Название SSL так сильно укоренилось в отрасли, что в большинстве случаев то, что называют SSL, скорее всего, является TLS. Например, OpenSSL поддерживает и SSL (версии 3.0), и TLS (версий 1.0, 1.1, 1.2 и 1.3).

Незащищенное соединение через HTTP и другие протоколы создает угрозу безопасности и целостности данных. Для их защиты используется комбинация SSL/TLS и протоколов уровня приложений. Например, HTTPS (HTTP over SSL) называется комбинация протокола HTTP и SSL, которая обеспечивает безопасное взаимодействие между Web-браузером и Web-сервером. Поддержка протокола HTTPS встроена во все современные браузеры. При использовании HTTPS адрес URL начинается с *https://*, а не с *http://*. Обычное соединение HTTP использует порт 80. Для соединения HTTPS определен порт 443.

HTTPS описан в RFC 2818 «HTTP Over TLS». Фундаментальных различий при использовании HTTP поверх SSL или TLS не существует. Обе реализации называются HTTPS и шифруют следующие элементы, передаваемые между браузером и сервером:

- URL запрашиваемого документа;
- содержимое документа;
- содержимое заполненных в браузере форм;
- куки, отправляемые браузером серверу и сервером браузеру;
- содержимое заголовка HTTP.

В устройствах D-Link протоколы SSL/TLS используются для безопасного подключения к Web-интерфейсу управления.

Следует отметить, что за протоколами SSL/TLS не закреплено какого-либо общеизвестного номера порта TCP/UDP. При их использовании

9. Протоколы уровня приложений

с протоколом более высокого уровня принято использовать номер порта защищенного варианта этого протокола. Например, для HTTPS общеизвестный номер порта — 443, для FTPS — 990, для TELNETS — 992, POP3S — 995.

9.3.1. Архитектура SSL/TLS

Протоколы SSL/TLS предоставляют приложениям, работающим поверх них, три основных сервиса: аутентификацию, конфиденциальность и целостность данных.

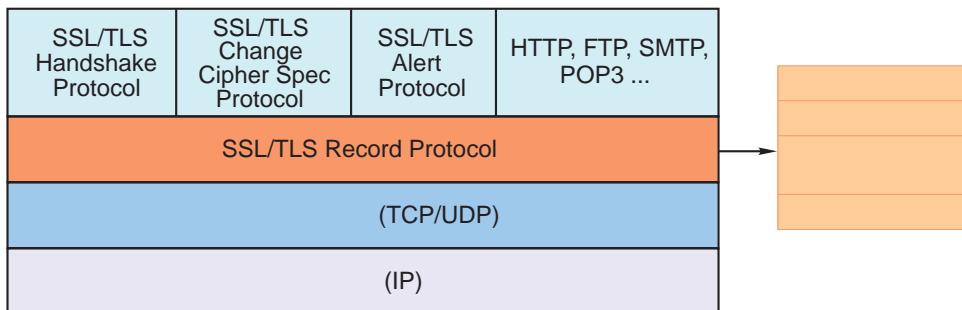


Рис. 9.12. Стек протоколов SSL/TLS

Стек протоколов SSL/TLS (рис. 9.12) состоит из двух уровней. На верхнем уровне расположены три протокола стека SSL/TLS: **Handshake** (протокол рукопожатия), **Change Cipher Spec** (протокол изменения параметров шифрования) и **Alert** (протокол уведомления об ошибке). Основным среди них является протокол Handshake. Он позволяет сторонам договориться о параметрах безопасности для протокола записи, аутентифицировать друг друга, получать разделяемый материал для создания ключей шифрования.

На нижнем уровне, расположенном над TCP, находится протокол **SSL/TLS Record** (протокол записи SSL/TLS). Его задачами являются обеспечение конфиденциальности и целостности данных. Протокол записи SSL/TLS используется для инкапсуляции данных различных протоколов уровня приложений (HTTP, FTP, TELNET, POP3 и т. д.), а также протоколов Handshake, Change Cipher Spec и Alert. Протокол записи принимает сообщение от приложения, фрагментирует данные на управляемые блоки, опционально сжимает их, вычисляет MAC (Message Authentication Code), шифрует, добавляет заголовок и передает на транспортный уровень. При получении данных он расшифровывает их, проверяет целостность, распаковывает, повторно собирает и доставляет приложению.

Протокол Handshake отвечает за установку сессии. *Сессия* — это логическое соединение между приложением клиента и сервера. Функция возобновления протокола Handshake позволяет создавать внутри одной сессии множество соединений. *Соединение* — это транспорт, который обеспечивает подходящий тип сервиса. Соединения могут совместно использовать набор

параметров безопасности, согласованных в процессе установки сессии. За счет этого снижаются вычислительные издержки на установку безопасного соединения между приложениями, т. е. не требуется каждый раз повторно согласовывать параметры безопасности.

Состояние соединения является рабочей средой протокола Record. Другими словами, протокол записи работает в соответствии с группой параметров, называемых состоянием соединения. Оно определяет алгоритмы сжатия, шифрования и хеширования, ключи шифрования. Существуют четыре состояния соединения: рабочие состояния чтения (current read) и записи (current write), ожидаемые состояния чтения (pending read) и записи (pending write). Под чтением понимается прием данных, под записью — передача данных. Различие между ожидаемым и рабочим состоянием следующее. Рабочим состоянием называются параметры безопасности, которые действуют в настоящий момент. Ожидаемое состояние — это группа параметров безопасности, предназначенная для будущего использования и устанавливаемая протоколом Handshake. Когда происходят изменения (рис. 9.13), ожидаемое состояние становится рабочим и создается новое пустое ожидаемое состояние.

При инициализации соединения рабочее состояние NULL, т. е. не определены ни ключи шифрования, ни алгоритмы сжатия, шифрования и хеширования. В это время начинает работу протокол Handshake, который создает параметры безопасности для ожидаемого состояния. Когда все готово, ожидаемое состояние становится рабочим, и параметры безопасности вступают в силу.

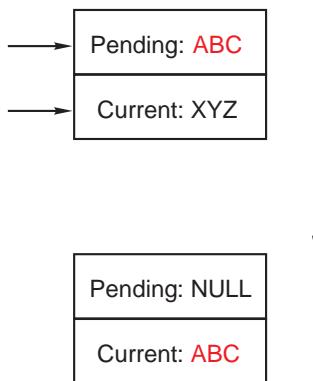


Рис. 9.13. Изменение состояния соединения

9.3.2. Основные отличия TLS 1.2 и TLS 1.3

Версия TLS 1.3 не является развитием предыдущих версий протокола. По сути она представляет собой новый протокол, который обеспечивает значительное повышение безопасности и производительности. Некоторое

время обсуждался вопрос, следует ли новую версию назвать TLS 2.0. В итоге остановились на названии TLS 1.3.

Кратко опишем основные отличия TLS 1.3 от TLS 1.2.

- Из списка поддерживаемых протоколов симметричного шифрования удалены все устаревшие алгоритмы (RC4, 3DES и др.). Остались только алгоритмы Authenticated Encryption with Associated Data (AEAD).

- Переопределен параметр *Cipher Suite*. В нем не указываются механизмы аутентификации и обмена ключами. Поэтому Cipher Suite TLS 1.3 не может быть использован для TLS 1.2 и, наоборот, даже при использовании одинакового состава криптографических алгоритмов.

- Добавлен новый режим Zero Round-Trip Time (0-RTT), который позволяет клиенту отправлять серверу зашифрованные данные в первом сообщении. Для аутентификации сервера и шифрования данных клиент использует PSK.

- Исключается использование RSA и фиксированного алгоритма Диффи — Хеллмана (DH) в качестве метода обмена ключами. Оставшиеся методы обеспечивают Forward Secrecy.

- Все сообщения протокола Handshake после Server Hello шифруются.

- Для генерации ключей используется функция HMAC-based Extract-and-Expand Key Derivation Function (HKDF).

- Машина состояний протокола Handshake значительно изменена. Удалены лишние сообщения, такие как Change Cipher Spec (за исключением случаев, когда необходима совместимость с промежуточными устройствами).

- Основными алгоритмами являются алгоритмы на эллиптических криевых. Добавлены новые алгоритмы цифровой подписи, такие как EdDSA.

- Удалены сжатие, алгоритм цифровой подписи Digital Signature Algorithm (DSA) и настраиваемые группы Ephemeral Diffie-Hellman (DHE).

9.3.3. Протокол Change Cipher Spec

Протокол изменения параметров шифрования (Change Cipher Spec) является самым простейшим из трех протоколов стека SSL/TLS. Он состоит из единственного сообщения Change Cipher Spec (рис. 9.14), которое содержит один байт со значением 1. Это сообщение отправляет и клиент,

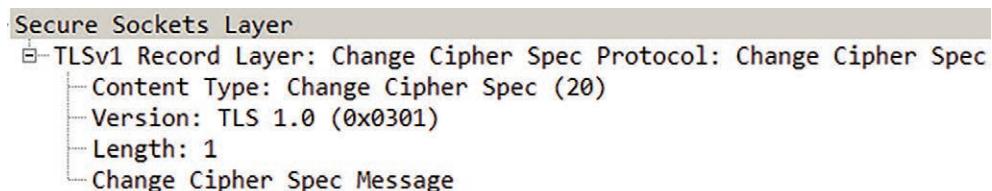


Рис. 9.14. Сообщение Change Cipher Spec

и сервер с целью уведомления получателя, что последующие сообщения протокола записи будут защищены новыми параметрами безопасности.

Получение этого сообщения заставляет протокол записи изменить ожидаемое состояние чтения (pending read) на рабочее состояние чтения (current read). Отправка этого сообщения заставляет протокол записи изменить ожидаемое состояние записи (pending write) на рабочее состояние записи (current write). Сообщение Change Cipher Spec отправляется в процессе работы протокола Handshake сразу после согласования параметров безопасности и до передачи сообщения Finished.

9.3.4. Протокол Alert

Протокол уведомления об ошибке (Alert) используется для передачи уведомлений об ошибке (error alert) (рис. 9.18) и закрытии соединения (closure alert) между клиентом и сервером. Уведомления шифруются протоколом записи в соответствии с рабочим состоянием.

Каждое сообщение протокола Alert состоит из двух байтов. Первый байт принимает значение 1 (warning) или 2 (fatal) для уведомления о серьезности ошибки. Если ошибка фатальная (fatal), SSL/TLS сразу же прерывает соединение. Другие соединения этой же сессии могут продолжать работу, но новые не должны устанавливаться. Если это предупреждение (warning), соединение может продолжать работать в обычном режиме. Второй байт содержит код, который определяет ошибку.

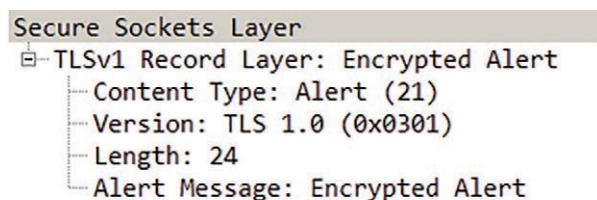


Рис. 9.15. Уведомление об ошибке decryption_failed_RESERVED

9.3.5. Сертификаты X.509

В протоколах SSL/TLS при установлении безопасного соединения применяются открытые ключи. Для их распространения используются сертификаты стандарта X.509 v3, связанные с каждым пользователем.

Рекомендация ITU-T X.509 является частью серии рекомендаций X.500, которые определяют сервисы директории. По сути директория — это сервер или распределенный набор серверов, который хранит базу данных с информацией о пользователях. Информация включает привязку имени пользователя к сетевому адресу, а также другие атрибуты. X.509 определяет структуру,

9. Протоколы уровня приложений

которая обеспечивает аутентификацию пользователей через сервисы директории X.500. Директория может служить хранилищем *сертификатов открытого ключа* (public-key certificates).

Каждый сертификат X.509 содержит открытый ключ пользователя и подписывается закрытым ключом доверенного *удостоверяющего центра* (Certification Authority, CA). Помимо того, сертификат может быть самоподписаным. Подпись необходима, чтобы убедиться, что данный сертификат не был подделан, и подтвердить, что данный открытый ключ является открытым ключом данного пользователя.

Удостоверяющий центр является доверенной «третьей стороной», которой доверяют владелец сертификата и сторона, полагающаяся на сертификат. Зачастую CA фактически принадлежит той же организации, что и конечные пользователи.

Под термином «корневой удостоверяющий центр» (root CA) подразумевается CA, которому непосредственно доверяет конечный пользователь. Этот CA не обязательно должен быть на вершине иерархии удостоверяющих центров.

Дополнительно X.509 определяет альтернативный протокол аутентификации, базирующийся на сертификатах открытого ключа. X.509 использует криптографию с открытым ключом и цифровую подпись.

Формат сертификата X.509 v3 включает следующие поля (RFC 5280):

- *Версия сертификата* (Version) — описывает версию сертификата;
- *Серийный номер* (Serial Number) — положительное целое число, назначаемое каждому сертификату удостоверяющим центром;
- *Алгоритм подписи* (Signature) — определяет алгоритм, используемый удостоверяющим центром для подписи сертификата;
- *Сведения об издателе* (Issuer) — определяет имя удостоверяющего центра, который подписал и выпустил сертификат;
- *Период действия сертификата* (Validity) — содержит даты начала и конца действия сертификата;
- *Сведения о владельце сертификата* (Subject) — содержит имя пользователя, связанного с открытым ключом, хранимым в поле Subject Public Key Info;
- *Информация об открытом ключе* (Subject Public Key Info) — содержит открытый ключ и определяет алгоритм, с которым используется ключ (RSA, DSA или Diffie-Hellman);
- *Уникальные идентификаторы* (Unique Identifiers) — необязательные поля, которые содержат уникальные идентификаторы удостоверяющего центра и владельца сертификата;
- *Расширения* (Extensions) — необязательное поле, которое содержит одно или несколько расширений сертификата.

Удостоверяющий центр подписывает сертификат своим закрытым ключом. Если соответствующий открытый ключ известен пользователю, то он

Технологии TCP/IP в современных компьютерных сетях

```
↳ X509Cert: Issuer: DigiCert SHA2 Extended Validation Server CA, www.digicert.com, DigiCert Inc, US, Subject: addons.mozilla.org, Cloud Services  
↳ SequenceHeader:  
↳ TbsCertificate: Issuer: DigiCert SHA2 Extended Validation Server CA, www.digicert.com, DigiCert Inc, US, Subject: addons.mozilla.org, Cloud Services  
↳ SequenceHeader:  
↳ Tag0:  
↳ Version: v3 (2)  
↳ SerialNumber: 0x09f9ee1fbf40f02d6f7e7891d1947968  
↳ Signature: Sha256WithRSAEncryption (1.2.840.113549.1.1.11)  
↳ Issuer: DigiCert SHA2 Extended Validation Server CA, www.digicert.com, DigiCert Inc, US  
↳ Validity: From: 09/28/17 00:00:00 UTC To: 10/04/19 12:00:00 UTC  
↳ Subject: addons.mozilla.org, Cloud Services, Mozilla Foundation, Mountain View, California, US, C2543436, Blob Value, Blob Value  
↳ SubjectPublicKeyInfo: RsaEncryption (1.2.840.113549.1.1.1)  
↳ Tag3:  
↳ Extensions:
```

Рис. 9.16. Сертификат X.509

может проверить, что подписанный удостоверяющим центром сертификат действителен.

Пользовательский сертификат, подписываемый удостоверяющим центром, имеет следующие характеристики:

- любой пользователь может прочитать сертификат и определить имя и открытый ключ владельца сертификата (рис. 9.17), срок действия сертификата;
- любой пользователь может удостовериться, что сертификат выпущен удостоверяющим центром и не является подделкой, а также удостоверить открытый ключ владельца сертификата;
- изменить или обновить сертификат может только удостоверяющий центр.

Сертификаты X.509 (рис. 9.16) могут формировать **цепочки доверия** (рис. 9.17), т. е. они могут быть подписаны одним или более промежуточными удостоверяющими центрами в иерархической манере.

Создание подписанного цифрового сертификата



Использование сертификата для проверки открытого ключа пользователя

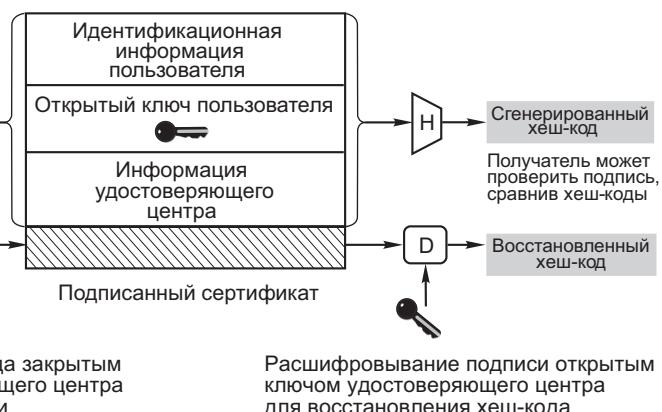


Рис. 9.17. Использование сертификата открытого ключа

9. Протоколы уровня приложений

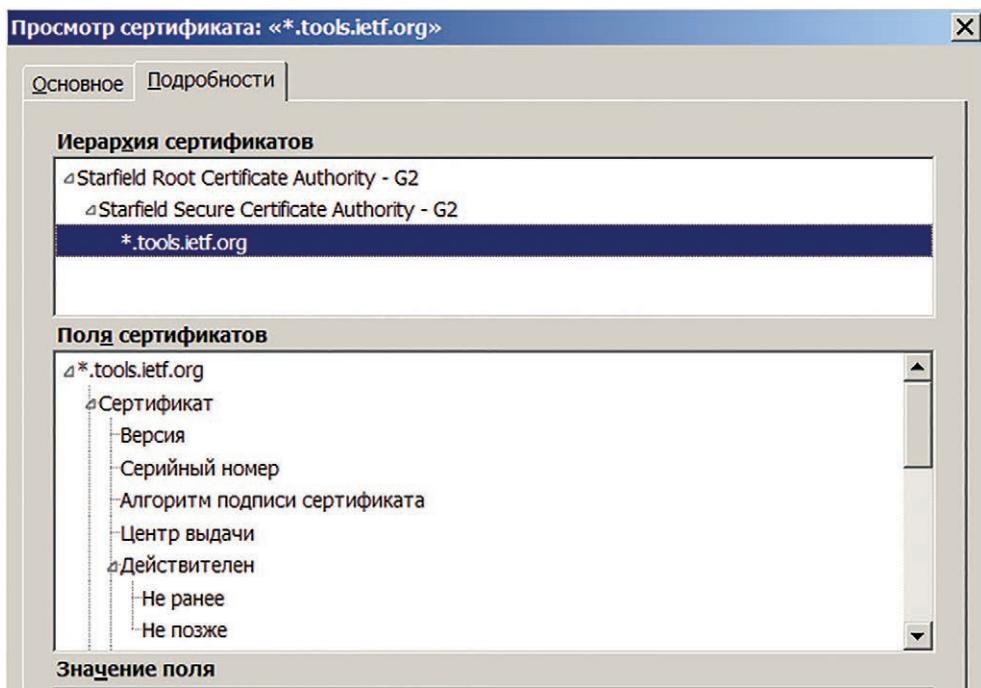


Рис. 9.18. Цепочка доверия сертификатов для tools.ietf.org

Также сертификат может быть подписан корневым СА напрямую.

В общем случае процесс получения сертификата (рис. 9.19) состоит из четырех шагов.

1. Первое звено в цепочке доверия — это удостоверяющий центр. В какой-то момент времени СА генерирует пару асимметричных ключей (закрытый и открытый) и с помощью закрытого ключа выпускает *самоподписанный сертификат* (поля *Issuer* и *Subject* данного сертификата X.509 совпадают). Этот сертификат является корневым сертификатом удостоверяющего центра. Закрытый ключ используется для подписания пользовательских сертификатов. Парный ему открытый ключ содержится в корневом сертификате. Корневые сертификаты (или сертификаты СА) распространяются среди конечных пользователей посредством какого-либо доверенного процесса, чаще всего при обновлении или установке программного обеспечения операционной системы или браузера. Каждая операционная система и большинство браузеров содержат список доверенных удостоверяющих центров.

2. Пользователь, желающий получить сертификат, рассматривает предложения различных удостоверяющих центров и выбирает среди них наиболее подходящий для него. В выбранный СА подается заявка на получение сертификата X.509. В зависимости от типа сертификата запрашивается и, как правило, проверяется различная информация о пользователе. Удостоверяющий

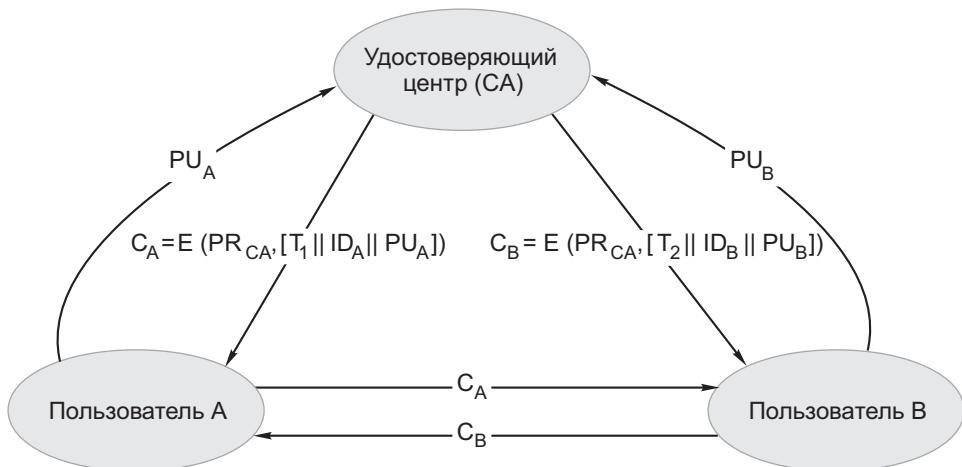
центр уполномочен достоверно установить, что пользователь является именно тем, за кого он себя выдает.

3. После предоставления необходимой идентификационной информации и проверки ее удостоверяющим центром пользователю предлагается сгенерировать пару асимметричных ключей (закрытый и открытый). С помощью закрытого ключа пользователь подписывает запрос на подписание сертификата (Certificate Signing Request, CSR), в котором наряду с прочей информацией содержится его открытый ключ. Формат CSR определен в RFC 5967.

4. CSR загружается на сервер удостоверяющего центра (обычно по FTP или HTTP). CA использует данные из CSR и, возможно, другую информацию, полученную из заявки, для создания сертификата X.509 пользователя. Наконец, CA подписывает сертификат пользователя, используя свой закрытый ключ. Сертификат X.509 отправляется пользователю (например, через FTP/HTTP/email).

Проверить подлинность сертификата может любой пользователь, имеющий доступ к открытому ключу CA, подписавшего сертификат. Как уже было сказано, открытый ключ содержится в корневом сертификате удостоверяющего центра, распространяемого в процессе какого-либо доверенного процесса.

В некоторых случаях пользователь может сам стать удостоверяющим центром и выпустить сертификат с помощью инструмента OpenSSL. Такой



C_A и C_B : сертификаты пользователей А и В;

PR_{CA} : закрытый ключ удостоверяющего центра;

T_1 и T_2 : временные метки;

ID_A и ID_B : идентификационная информация пользователей А и В;

PU_A и PU_B : открытые ключи пользователей А и В.

Рис. 9.19. Получение пользователями сертификатов открытого ключа и обмен ими

9. Протоколы уровня приложений

сертификат также называется самоподписаным. Данная форма самоподписанных сертификатов может быть использована либо во время тестирования, либо в рабочей среде, когда пользователь хочет организовать контроль доступа к какому-либо устройству в частной сети. В этом случае отношения доверия, обычно ассоциированные с внешним (официально признанным) CA, могут рассматриваться как неявные.

Когда программное обеспечение с поддержкой SSL/TLS (например, браузер) впервые встречает такой самопод подписанный сертификат и подразумевается, что у браузера нет копии соответствующего корневого сертификата, будет сгенерировано сообщение, спрашивающее пользователя, желает ли он принять сертификат. С другой стороны, самопод подписанный корневой сертификат может быть предустановлен или импортирован на компьютер пользователя, в этом случае браузер не будет генерировать сообщения об ошибке.

9.3.6. Протокол Handshake в TLS 1.2

Прежде чем клиент и сервер начнут обмениваться данными приложения через безопасное соединение SSL/TLS, они должны договориться об используемых параметрах безопасности: согласовать версию протокола SSL/TLS (рис. 9.20), алгоритмы шифрования, проверки целостности, криптографические

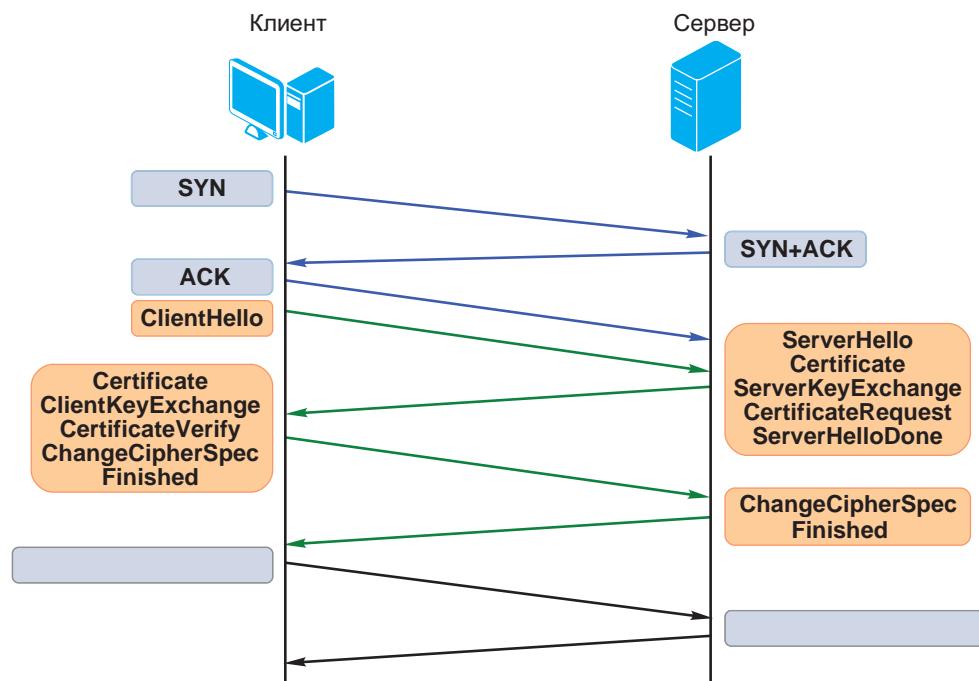


Рис. 9.20. Операции протокола SSL/TLS 1.2 Handshake

ключи и дополнительно аутентифицировать друг друга. Эти процессы выполняются протоколом Handshake. Его работа состоит из обмена серией сообщений между клиентом и сервером.

Функционирование протокола Handshake в TLS 1.2 и TLS 1.3 существенно различается. В данном разделе будет кратко описана работа протокола в TLS 1.2.

1. Согласование алгоритмов безопасности и аутентификация сервера.

Клиент отправляет серверу сообщение *Client Hello* (рис. 9.21), в ответ на которое сервер должен отправить свое сообщение *Server Hello*, или сообщение об ошибке, если она произошла. Сообщения Hello используются клиентом и сервером для установления следующих параметров сессии: версии протокола (Protocol Version), идентификатора сессии (Session ID), набора алгоритмов шифрования (Cipher Suite) и метода сжатия (Compression Method). Дополнительно клиент и сервер генерируют два случайных 32-байтных значения *ClientHello.random* и *ServerHello.random* и обмениваются ими. Позднее они будут использоваться для вычисления общего секретного ключа.

Для сообщения *Server Hello* (рис. 9.22) применяются следующие соглашения. Поле *Version* содержит наименьшую из версий, предложенных клиентом, и наибольшую, из поддерживаемых сервером. Значение *ServerHello.random* в поле *Random* генерируется сервером и не зависит от *ClientHello.random*. Если поле *Session ID* клиента было не равно нулю, это же значение используется сервером; в противном случае поле *Session ID* сервера содержит значение идентификатора для новой сессии. Поле *Cipher Suite* содержит единственный набор алгоритмов шифрования, выбранный сервером из предложенных клиентом. Поле *Compression Method* содержит метод сжатия, выбранный сервером среди предложенных клиентом. Обычно это NULL.

Первым компонентом параметра *Cipher Suite* сообщения *Hello* является метод обмена ключами. В SSL/TLS 1.2 поддерживаются перечисленные ранее методы.

- **RSA.** Открытый ключ содержится в сертификате сервера. Секретный ключ шифруется клиентом с помощью открытого ключа сервера.

• **Фиксированный алгоритм Диффи — Хеллмана (DH).** Обмен ключами Диффи — Хеллмана, при котором сертификат сервера содержит открытые параметры, подписанные удостоверяющим центром (CA), что позволяет клиенту проверить их подлинность. Это означает, что параметры фиксированы от сессии к сессии. Клиент предоставляет свой открытый ключ Диффи — Хеллмана или в сертификате, если требуется его аутентификация, или в сообщении обмена ключами. Результатом работы этого метода является вычисление на основе фиксированных открытых ключей общего для клиента и сервера фиксированного секретного ключа.

• **Эфемерный алгоритм Диффи — Хеллмана (DHE).** Этот метод используется для создания временных (одноразовых) секретных ключей. Для

9. Протоколы уровня приложений

```
> Transmission Control Protocol, Src Port: 49181, Dst Port: 443, Seq: 1, Ack: 1, Len: 168
✓ Secure Sockets Layer
  ✓ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 163
  ✓ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 159
    Version: TLS 1.2 (0x0303)
  ✓ Random
    GMT Unix Time: Apr 15, 2019 01:14:55.000000000 RTZ 2 (зима)
    Random Bytes: c53b77c7ba9063652ddc2a54518300eb477bbdb707eac5de...
    Session ID Length: 0
    Cipher Suites Length: 30
  ✓ Cipher Suites (15 suites)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa8)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
    Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
  Compression Methods Length: 1
  ✓ Compression Methods (1 method)
    Compression Method: null (0)
  Extensions Length: 88
  > Extension: Extended Master Secret
  > Extension: renegotiation_info
  > Extension: elliptic_curves
  > Extension: ec_point_formats
  > Extension: SessionTicket TLS
  > Extension: Application Layer Protocol Negotiation
  > Extension: status_request
  > Extension: signature_algorithms
```

Рис. 9.21. Сообщение Client Hello

каждой сессии сервер генерирует новый открытый ключ — Диффи — Хеллмана и подписывает его своим закрытым ключом RSA или DSA (Digital Signature Algorithm). Клиент может использовать соответствующий открытый ключ для проверки подписи. Для аутентификации открытых ключей используются сертификаты.

```
> Transmission Control Protocol, Src Port: 443, Dst Port: 49181, Seq: 1, Ack: 169, Len: 79
└ Secure Sockets Layer
  └ TLSv1 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 74
  └ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 70
    Version: TLS 1.0 (0x0301)
  └ Random
    GMT Unix Time: Jan 1, 1970 12:00:41.000000000 RTZ 2 (зима)
    Random Bytes: 0000762500003bc2000052f200001cc7000062b900004a67...
    Session ID Length: 32
    Session ID: 00003cf30000338300003b44000064eb0000312600000edc...
    Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
    Compression Method: null (0)
```

Рис. 9.22. Сообщение Server Hello

• **Анонимный алгоритм Диффи — Хеллмана.** Базовый алгоритм Диффи — Хеллмана используется без аутентификации. Каждая сторона отправляет открытые параметры Диффи — Хеллмана другой стороне без аутентификации. Этот подход уязвим для атак типа «человек посередине».

Вслед за методом обмена ключами в параметре *Cipher Suite* указана спецификация алгоритмов шифрования *CipherSpec*, включающая два поля:

- алгоритмы шифрования: RC4, RC2, DES, 3DES, DES40, IDEA, AES и др.,
- алгоритмы хеширования: MD5, SHA-1.

Если необходима аутентификация сервера, то вслед за сообщением *Server Hello* в сообщении *Certificate* (рис. 9.23) он отправляет свой сертификат или цепочку сертификатов X.509.v3. Отправка сертификата требуется для любого согласованного метода обмена ключами, за исключением анонимного алгоритма Диффи — Хеллмана. При использовании фиксированного алгоритма Диффи — Хеллмана сертификат содержит открытый ключ Диффи — Хеллмана и выполняет функцию сообщения обмена ключами.

Если у сервера нет сертификата или сертификат используется только для подписи, он отправляет сообщение обмена ключами *Server Key Exchange*. Это сообщение не требуется в следующих случаях: 1) сервер отправил сертификат с фиксированными параметрами Диффи — Хеллмана или 2) используется метод обмена ключами RSA.

Далее не анонимный сервер (сервер, не использующий анонимный алгоритм Диффи — Хеллмана) может запросить сертификат у клиента. Сообщение *Certificate Request* включает два поля: *certificate_types* и *certificateAuthorities*. Поле *certificate_types* содержит список запрашиваемых типов сертификатов, упорядоченных в порядке их предпочтения сервером. Поле *certificateAuthorities* содержит список полученных из X.509 доверенных удостоверяющих центров.

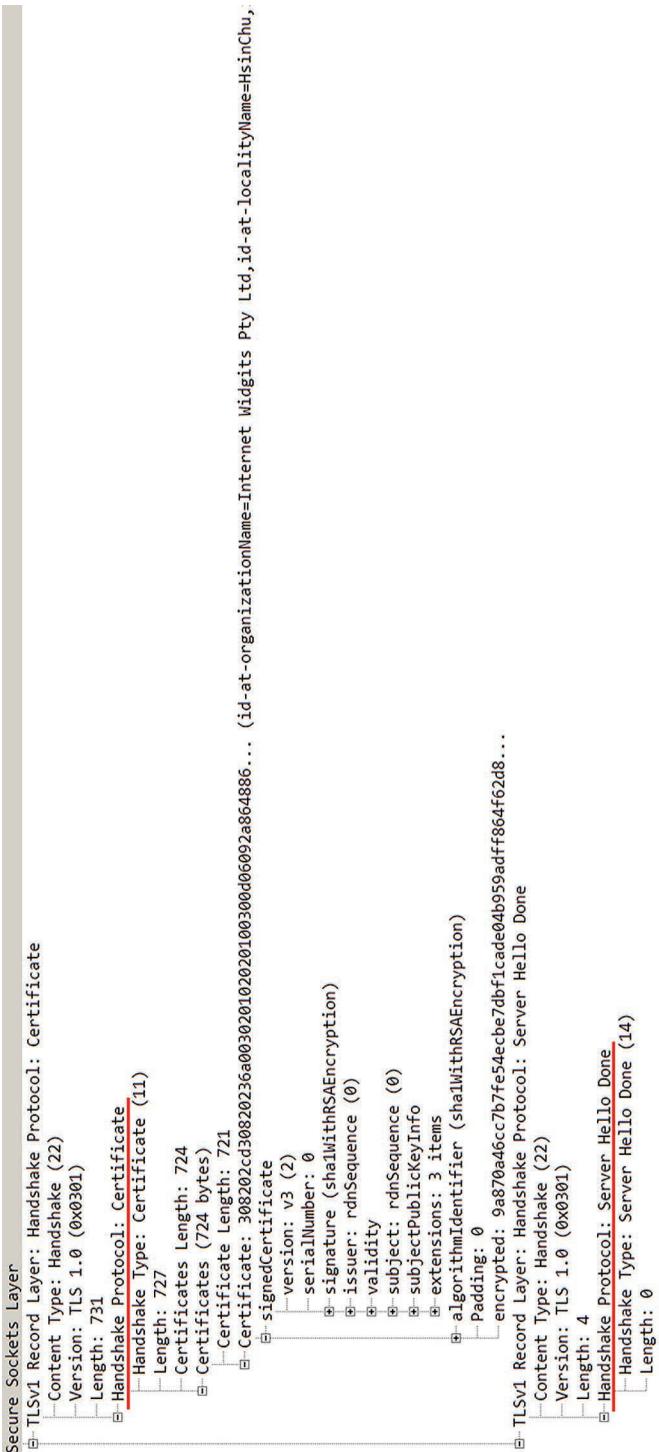


Рис. 9.23. Сообщения Certificate, Server Hello Done

Финальным сообщением, которое всегда отправляет сервер, является Server Hello Done (см. рис. 9.23). Оно не имеет параметров и сообщает о конце фазы обмена сообщениями Hello. После отправки этого сообщения сервер ожидает ответ клиента.

2. Аутентификация клиента и обмен ключами.

После получения сообщения Server Hello Done клиент должен удостовериться, что сервер предоставил действительный сертификат (при необходимости) и проверить приемлемость параметров сообщения. Если все они приемлемы, клиент отправляет серверу одно или несколько сообщений. Если сервер запросил сертификат, клиент должен отправить ему или сообщение Certificate, или, если нет подходящего сертификата, уведомление no_certificate.

Далее клиент отправляет сообщение Client Key Exchange. Его содержимое зависит от алгоритма обмена ключами, выбранного клиентом и сервером в процессе обмена сообщениями Hello, следующим образом.

- **RSA.** Клиент генерирует 48-байтный ключ pre_master_secret, шифрует его открытым ключом из сертификата сервера и отправляет результат в зашифрованном сообщении premaster secret.

- **Фиксированный алгоритм Диффи — Хеллмана.** Отправляются открытые параметры Диффи — Хеллмана, которые позволяют сторонам вычислить общий pre_master_secret. Если открытые параметры Диффи — Хеллмана были отправлены клиентом в сообщении Certificate, содержимое этого сообщения нулевое.

```
Secure Sockets Layer
└─ TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
    └─ Content Type: Handshake (22)
    └─ Version: TLS 1.0 (0x0301)
    └─ Length: 134
    └─ Handshake Protocol: Client Key Exchange
        └─ Handshake Type: Client Key Exchange (16)
        └─ Length: 130
        └─ RSA Encrypted PreMaster Secret
            └─ Encrypted PreMaster length: 128
            └─ Encrypted PreMaster: a3e10d682460ab367dd08933a356dffdebcc8acdde4a7b65...
└─ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    └─ Content Type: Change Cipher Spec (20)
    └─ Version: TLS 1.0 (0x0301)
    └─ Length: 1
    └─ Change Cipher Spec Message
└─ TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
    └─ Content Type: Handshake (22)
    └─ Version: TLS 1.0 (0x0301)
    └─ Length: 40
    └─ Handshake Protocol: Encrypted Handshake Message
```

Рис. 9.24. Сообщения Client Key Exchange, Change Cipher Spec, Finished

- **Эфемерный или анонимный алгоритм Диффи — Хеллмана.** Отправляются открытые параметры Диффи — Хеллмана.

Клиент может отправить сообщение Certificate Verify, которое используется для точной проверки его сертификата. Это сообщение посыпается только вслед за клиентским сертификатом, имеющим возможность подписи (т. е. вслед за всеми сертификатами, за исключением тех, которые содержат параметры фиксированного алгоритма Диффи — Хеллмана).

3. Окончание обмена.

Эта фаза завершается установлением безопасного соединения. Клиент посыпает сообщение Change Cipher Spec (рис. 9.25) и копирует ожидаемые параметры безопасности в рабочие. Это сообщение не является частью протокола Handshake и отправляется протоколом Change Cipher Spec.

Сразу после отправки Change Cipher Spec, клиент посыпает шифрованное сообщение Finished, содержащее новые алгоритмы, ключи и секреты. Оно подтверждает, что обмен ключами и аутентификация выполнены успешно.

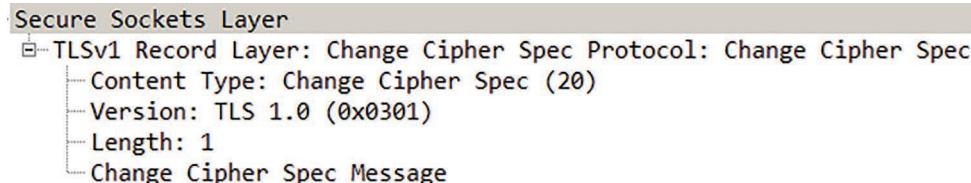


Рис. 9.25. Сообщение Change Cipher Spec сервера

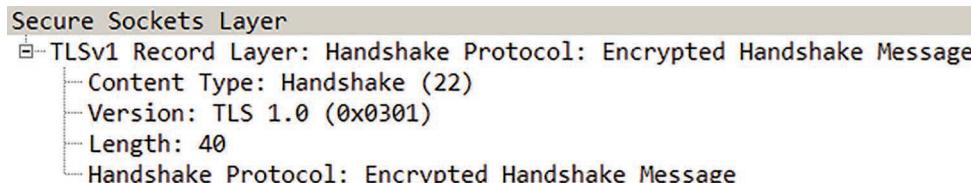


Рис. 9.26. Шифрованное сообщение Finished сервера

В ответ на эти два сообщения сервер отправляет свое сообщение Change Cipher Spec, копирует ожидаемые параметры безопасности в рабочие и посыпает сообщение Finished (рис. 9.26) с новыми параметрами.

Сообщение Finished не требует подтверждения. При его приеме получатель должен убедиться в правильности всех параметров. Как только стороны соединения отправили друг другу свои сообщения Finished, получили их и проверили параметры, они могут начать обмен шифрованными данными приложений (рис. 9.27). На этом работа протокола рукопожатия завершается.

```
Secure Sockets Layer
└─ TLSv1 Record Layer: Application Data Protocol: http-over-tls
    ├ Content Type: Application Data (23)
    ├ Version: TLS 1.0 (0x0301)
    └ Length: 24
    └ Encrypted Application Data: e9757b2a4f6141f5e9eebdd95e12db36af02d19054f3390
└─ TLSv1 Record Layer: Application Data Protocol: http-over-tls
    ├ Content Type: Application Data (23)
    ├ Version: TLS 1.0 (0x0301)
    └ Length: 392
    └ Encrypted Application Data: 8be96b9504957a184f452cc02fef3b5927d8667f20fd3c96...
```

Рис. 9.27. Шифрованные данные приложения

Криптографические вычисления

Давайте рассмотрим, как в SSL/TLS 1.2 вычисляется разделяемый секретный мастер-ключ (`master_secret`) и как из него генерируются криптографические параметры для протокола записи.

Разделяемый секретный мастер-ключ — это 48-байтное значение, генерируемое для данной сессии посредством защищенного обмена ключами. Его вычисление состоит из двух стадий. Сначала стороны обмениваются предварительным мастер-ключом (`pre_master_secret`), а далее вычисляют мастер-ключ.

Обмен предварительным мастер-ключом может выполняться следующим образом.

- **RSA.** Клиент генерирует 48-байтный ключ `pre_master_secret`, шифрует его открытым ключом RSA сервера и отправляет ему результат в шифрованном сообщении `premaster secret`. Сервер расшифровывает сообщение с помощью своего закрытого ключа и восстанавливает `pre_master_secret`.

- **Диффи — Хеллман.** Клиент и сервер генерируют открытые ключи Диффи — Хеллмана. После обмена ими каждая сторона вычисляет общий `pre_master_secret`.

Все методы обмена ключами используют одинаковый метод преобразования предварительного мастер-ключа в секретный мастер-ключ.

В SSL 3.0 секретный мастер-ключ вычисляется так:

```
master_secret =
  MD5(pre_master_secret + SHA('A' + pre_master_secret +
    ClientHello.random + ServerHello.random)) +
  MD5(pre_master_secret + SHA('BB' + pre_master_secret +
    ClientHello.random + ServerHello.random)) +
  MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
    ClientHello.random + ServerHello.random));
```

9. Протоколы уровня приложений

В TLS 1.2 вычисление секретного мастер-ключа отличается от вычисления SSL 3.0. Алгоритм выполняется до тех пор, пока не будет получено 48-байтное случайное число:

```
master_secret = PRF(pre_master_secret, "master secret",
                     ClientHello.random + ServerHello.random)
                     [0..47];
```

`ClientHello.random` и `ServerHello.random` — два случайных 32-байтных значения, которыми клиент и сервер обменялись в сообщениях Hello.

Протокол записи требует, чтобы ключи для рабочего состояния соединения генерировались из параметров безопасности, предоставленных протоколом рукопожатия. Из секретного мастер-ключа последовательно генерируются следующие ключи:

- *Client write MAC secret* (секретный ключ MAC клиента): секретный ключ, используемый при вычислении MAC над данными, отправленными клиентом;
- *Server write MAC secret* (секретный ключ MAC сервера): секретный ключ, используемый при вычислении MAC над данными, отправленными сервером;
- *Client write key* (ключ шифрования клиента): секретный ключ, используемый для шифрования данных клиентом и расшифровывания их сервером;
- *Server write key* (ключ шифрования сервера): секретный ключ, используемый для шифрования данных сервером и расшифровывания их клиентом.
- *Client/Server write Initialization Vectors* (векторы инициализации клиента/сервера): при использовании блочного шифрования в режиме Cipher Block Chaining (CBC) для каждого ключа поддерживается вектор инициализации (IV).

В SSL 3.0 ключи генерируются из секретного мастер-ключа путем его хеширования в последовательность байтов длиной, достаточной для формирования всех необходимых параметров:

```
key_block = MD5(master_secret + SHA(`A' + master_secret +
                                         + ServerHello.random +
                                         + ClientHello.random)) +
            MD5(master_secret + SHA(`BB' + master_secret +
                                         + ServerHello.random +
                                         + ClientHello.random)) +
            MD5(master_secret + SHA(`CCC' + master_secret +
                                         + ServerHello.random +
                                         + ClientHello.random)) + [...];
```

В TLS 1.2 ключи из секретного мастер-ключа вычисляются с использованием псевдослучайной функции. Вычисления продолжаются до тех пор, пока не будет получена последовательность байтов достаточной длины, из которой будут сформированы необходимые ключи:

```
key_block = PRF(SecurityParameters.master_secret,
                 "key expansion",
                 SecurityParameters.server_random +
                 SecurityParameters.client_random);
```

9.3.7. Протокол Handshake в TLS 1.3

TLS 1.3 поддерживает три основных метода обмена ключами:

- (EC)DHE (алгоритм Диффи — Хеллмана на поле конечного размера и эллиптических кривых);
- PSK (Pre-Shared Key);
- PSK с (EC)DHE.

Протокол Handshake в TLS 1.3 состоит из трех фаз (рис. 9.28):

1. **Обмен ключами:** согласовывается разделяемый материал для создания ключей и выбираются криптографический параметры. Все сообщения, передаваемые после этой фазы, шифруются.

2. **Параметры сервера:** согласовываются различные параметры, отличные от предыдущей фазы (необходимость аутентификации клиента и др.).

3. **Аутентификация:** аутентификация сервера (опционально и клиента), подтверждение ключей и целостности обмена.

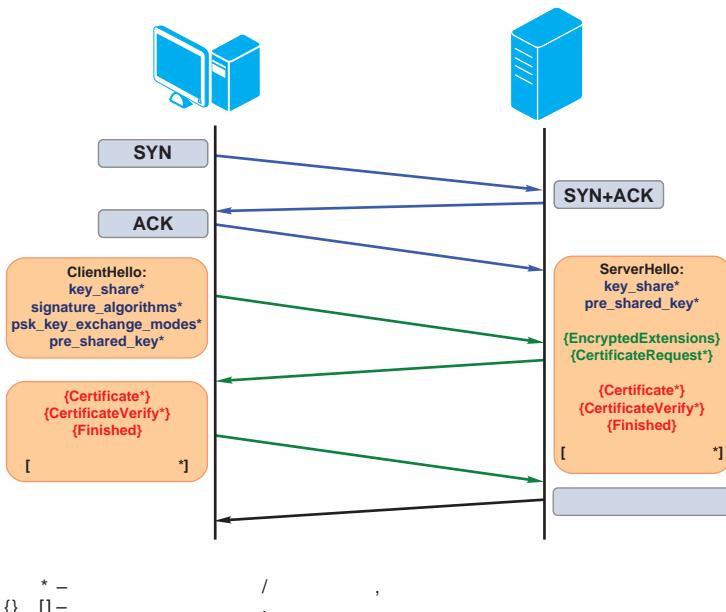


Рис. 9.28. Операции полного обмена протокола TLS 1.3 Handshake

По сравнению с TLS 1.2 обмен в TLS 1.3 короче на одну итерацию (половину цикла двухстороннего обмена сообщениями). В TLS 1.2 клиент и сервер могут начать передачу данных только после обмена сообщениями `Finished`. В TLS 1.3 сервер начинает передачу данных приложения раньше, не дожидаясь получения от клиента сообщения `Finished`.

На рис. 9.28 в фигурных и квадратных скобках показаны сообщения и расширения, которые передаются в зашифрованном виде. Все сообщения протокола Handshake после `Server Hello` шифруются. Для защиты сообщений протокола Handshake и данных приложения используются отдельные ключи.

Кратко опишем операции, выполняемые в каждой фазе обмена.

Сообщения обмена ключами `Hello` используются для определения функций безопасности, поддерживаемых клиентом и сервером, а также создания разделяемых секретных ключей.

Начинает обмен клиент. Он отправляет сообщение `Client Hello` (рис. 9.29), которое содержит:

- случайное число `ClientHello.random`;
- набор алгоритмов шифрования *Cipher Suite* (рис. 9.30);
- поддерживаемые версии протокола TLS в расширении `supported_versions`;
- поддерживаемые группы (EC)DHE в расширении `supported_groups` и открытые ключи Диффи — Хеллмана в расширении `key_share` для некоторых или всех групп;
- список идентификаторов симметричных ключей известных клиенту в расширении `pre_shared_key` и режим обмена ключами, используемый с PSK в расширении `psk_key_exchange_modes`;
- поддерживаемые алгоритмы цифровой подписи в расширении `signature_algorithms`;
- дополнительные расширения.

В поле *Cipher Suite* клиент отправляет список поддерживаемых криптографических алгоритмов. В TLS 1.3 параметр *Cipher Suite* включает два компонента: алгоритм шифрования с указанием длины ключа и алгоритм хеширования. В отличие от TLS 1.2 спецификация TLS 1.3 на данный момент (2018) определяет всего пять наборов алгоритмов шифрования:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_AES_128_CCM_SHA256`
- `TLS_AES_128_CCM_8_SHA256`

Правила именования делают параметр *Cipher Suite* TLS 1.3 несовместимым с TLS 1.2. Однако клиент может передавать *Cipher Suite* и для TLS 1.2, и для TLS 1.3. Это возможно благодаря тому, что формат идентификатора набора, который указывается в конце поля *Cipher Suite*, не изменился и равен 2 байтам. Понятно, что сервер с поддержкой TLS 1.3 выберет один из наборов, соответствующих этой спецификации.

```
└─ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    ├ Content Type: Handshake (22)
    ├ Version: TLS 1.0 (0x0301)
    └ Length: 512
└─ Handshake Protocol: Client Hello
    ├ Handshake Type: Client Hello (1)
    └ Length: 508
    Version: TLS 1.2 (0x0303)
    Random: b90fd80253fe3bdc609d60d191b464adce6aaa69684d23af...
    Session ID Length: 32
    Session ID: 1322fafc4f4d8864c461c19223d73be8b620ca339be80bba...
    Cipher Suites Length: 36
    ┌─ Cipher Suites (18 suites)
    └─ Compression Methods Length: 1
    ┌─ Compression Methods (1 method)
    └─ Extensions Length: 399
    ┌─ Extension: server_name (len=17)
    ┌─ Extension: extended_master_secret (len=0)
    ┌─ Extension: renegotiation_info (len=1)
    ┌─ Extension: supported_groups (len=14)
    ┌─ Extension: ec_point_formats (len=2)
    ┌─ Extension: SessionTicket TLS (len=0)
    ┌─ Extension: application_layer_protocol_negotiation (len=14)
    ┌─ Extension: status_request (len=5)
    ┌─ Extension: Unknown type 51 (len=107)
    ┌─ Extension: supported_versions (len=9)
    ┌─ Extension: signature_algorithms (len=24)
    ┌─ Extension: psk_key_exchange_modes (len=2)
    ┌─ Extension: Unknown type 28 (len=2)
    └─ Extension: padding (len=146)
```

Рис. 9.29. Сообщение Client Hello TLS 1.3

Сервер обрабатывает сообщение Client Hello и определяет соответствующие криптографические параметры соединения. В ответ он может отправить сообщение Server Hello (рис. 9.31), сообщение HelloRetryRequest или сообщение об ошибке, если она произошла.

Если сервер может согласовать набор параметров, предложенных клиентом, он посыпает сообщение Server Hello, которое содержит:

- случайное число *ServerHello.random*;
- единственный набор алгоритмов шифрования, выбранный сервером из предложенных клиентом (в поле *Cipher Suite*);
- список расширений. В него включаются только те расширения, которые позволяют согласовать криптографический контекст и версию протокола.

9. Протоколы уровня приложений

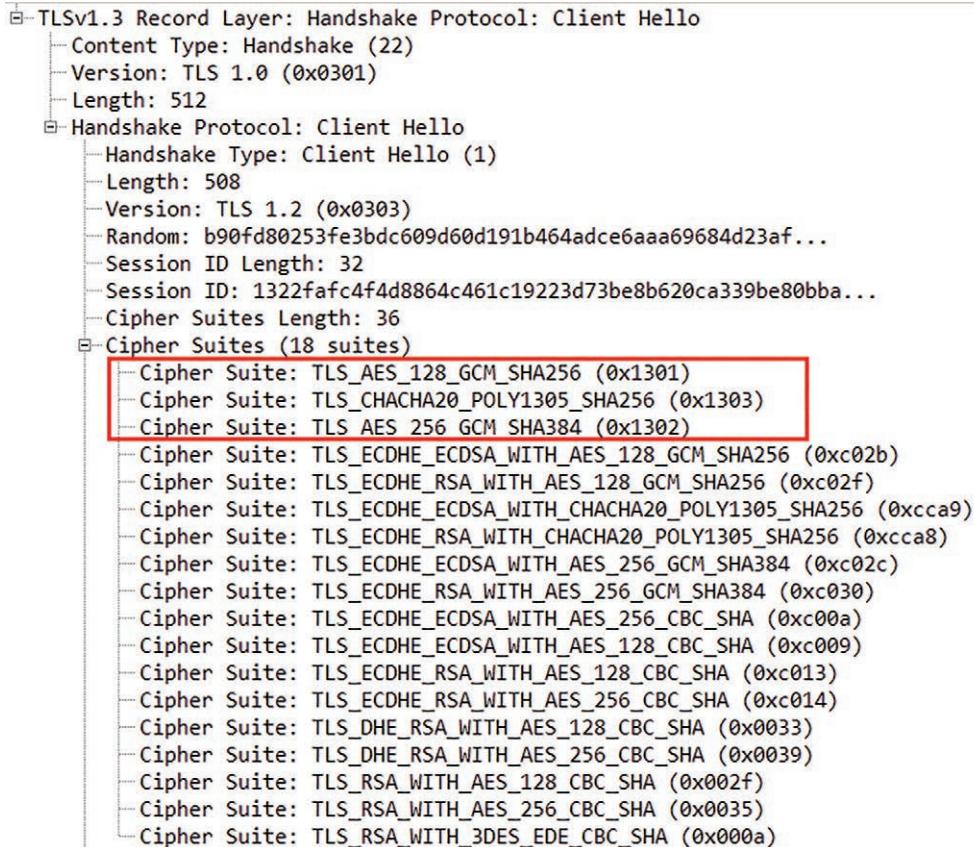


Рис. 9.30. Наборы Cipher Suite в сообщении Client Hello TLS 1.3

Все сообщения Server Hello TLS 1.3 должны содержать расширение supported_versions.

В TLS 1.3 запрещено повторное согласование соединения. Если клиент внутри открытой сессии отправит еще одно сообщение Client Hello, сервер прервет соединение и пришлет уведомление «unexpected_message».

Если сервер может выбрать приемлемый набор параметров, но клиент не предоставил достаточной информации для продолжения переговоров, он отправляет сообщение Hello Retry Request. Оно имеет такой же формат, как и Server Hello. В сообщение включается минимальный набор расширений, позволяющий клиенту правильно сгенерировать повторное сообщение Client Hello. Сервер может включить в сообщение расширение Cookie, чтобы проверить, что клиент действительно отвечает по указанному адресу. Это позволяет бороться с некоторыми видами DoS-атак.

```
└─ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    ├ Content Type: Handshake (22)
    ├ Version: TLS 1.2 (0x0303)
    └ Length: 122
    └─ Handshake Protocol: Server Hello
        ├ Handshake Type: Server Hello (2)
        ├ Length: 118
        ├ Version: TLS 1.2 (0x0303)
        ├ Random: 2177f80e2a78787598aa94a302c03e4d404f96a84a792575...
        ├ Session ID Length: 32
        ├ Session ID: 1322fafc4f4d8864c461c19223d73be8b620ca339be80bba...
        ├ Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        ├ Compression Method: null (0)
        └ Extensions Length: 46
        └─ Extension: supported_versions (len=2)
            ├ Type: supported_versions (43)
            ├ Length: 2
            └ Supported Version: TLS 1.3 (0x0304)
        └─ Extension: Unknown type 51 (len=36)
            ├ Type: Unknown (51)
            ├ Length: 36
            └ Data: 001d0020001017aa137cb394477fbf0fa7c00d69c39e2b2d...
    └─ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    └─ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
```

Рис. 9.31. Сообщение Server Hello TLS 1.3

Если в процессе одной сессии клиент дважды получит сообщение HelloRetry Request, он прервет соединение.

Комбинация параметров Client Hello и Server Hello позволяет определять разделяемые секреты. Если используется обмен по протоколу (EC)DHE, клиент передает открытые ключи Диффи — Хеллмана в расширении key_share. При условии, что поддерживаемые группы (EC)DHE у них совпадают, сервер выбирает требуемые параметры и присыпает клиенту свои открытые ключи Диффи — Хеллмана в расширении key_share.

Если используется обмен ключами на основе PSK (предварительно установленных ключей), клиент в расширении pre_shared_key передает список идентификаторов симметричных ключей. Это связано с тем, что ключи секретные и их нельзя передавать по открытому каналу. Из предложенных клиентом идентификаторов сервер выбирает подходящий и возвращает его клиенту в расширении pre_shared_key. Также сервер должен выбрать режим установки ключа из предложенных клиентом в расширении psk_key_exchange_modes. В настоящее время поддерживаются режимы PSK и PSK with (EC)DHE. В аналогичном расширении сервер сообщает о выбранном режиме.

9. Протоколы уровня приложений

Следует отметить, что (EC)DHE и PSK могут использоваться как по отдельности, так и вместе.

На данном этапе фаза обмена ключами завершается. Клиент и сервер генерируют ключи `client_handshake_traffic_secret` и `server_handshake_traffic_secret`.

Во второй фазе сервер отправляет два сообщения — `Encrypted Extensions` и `Certificate Request`. Оба сообщения шифруются с помощью ключей, полученных из `server_handshake_traffic_secret`. Сообщение `Encrypted Extensions` содержит расширения, которые могут быть защищены. В сообщении `Certificate Request` сервер может запросить сертификат клиента.

В третью фазе происходит аутентификация клиента и сервера. Она представлена сообщениями `Certificate`, `CertificateVerify` и `Finished`. Сообщения шифруются с помощью ключей, полученных из `client_handshake_traffic_secret` и `server_handshake_traffic_secret`. Сообщение `Finished` передается всегда. После его отправки можно начинать передачу данных приложения.

Сообщения `Certificate` и `CertificateVerify` передаются при определенных условиях. Как и в предыдущих версиях TLS, аутентификация может быть полностью исключена (анонимный режим), однако типичный сценарий использования подразумевает аутентификацию сервера клиентом. Передача сообщения `Certificate` сервером требует передачи и сообщения `CertificateVerify`. Сообщение `CertificateVerify` выполняет две важные задачи: позволяет клиенту проверить, что у сервера действительно есть секретный ключ от предоставленного сертификата; подтверждает целостность обмена до данной точки включительно.

Сообщение `Certificate Verify` состоит из идентификатора алгоритма подписи и самой подписи. Подписываются все сообщения `Handshake`, полученные и отправленные сервером к настоящему моменту.

Клиент проверяет подпись с помощью открытого ключа, содержащегося в сообщении `Certificate` сервера.

Как упоминалось ранее, в протоколе `Handshake` TLS 1.3 удалены лишние сообщения, такие как `Change Cipher Spec`. Однако на практике они передаются, но никак не учитываются клиентом и сервером. Сообщение `Change Cipher Spec` требуется для совместимости с промежуточными устройствами, такими как системы Deep Packet Inspection (DPI), чтобы они не разрывали соединение.

Согласование версии протокола

Протокол TLS обеспечивает встроенный механизм согласования версии между конечными точками.

TLS 1.0, TLS 1.1, TLS 1.2 и SSL 3.0 используют совместимые сообщения `Client Hello`. Поле версии протокола записи в них используется для разных целей.

В TLS 1.3 информация о поддерживаемых версиях передается в расширении `supported_versions` сообщений `Hello`. Поля, содержащие номер версии в предыдущих версиях протокола, сохранены в статусе исторических, их значения зафиксированы и не используются. Наличие расширения `supported_versions` является одним из признаков того, что клиент или сервер будут использовать версию TLS 1.3. В сообщении `Client Hello` расширение `supported_versions` содержит список версий протокола, поддерживаемых клиентом. Сервер включает в это расширение номер выбранной версии.

Режим 0-RTT

При проектировании TLS 1.3 большое внимание уделялось ускорению работы протокола. Основные временные затраты происходят при установлении соединения. Когда клиент и сервер используют PSK (полученный до установления соединения), TLS 1.3 позволяет клиенту отправлять данные приложения в первом же сообщении (так называемые «ранние данные» (*early data*)). Клиент использует ключ PSK для аутентификации сервера и шифрования ранних данных. Этот режим называется **0-RTT**. Он отличается от полной (1-RTT) схемы установления соединения только тем, что в первое сообщение добавляются данные приложения. Все остальные сообщения остаются без изменений.

Данный вариант установления соединения обладает рядом недостатков. При передаче ранних данных не обеспечивается прогрессивная секретность, так как они шифруются не сеансовым ключом, а предложенным ключом PSK. Не гарантируется защита от replay-аттак. Остальные данные будут защищены сеансовыми ключами. Повторная передача ранних данных не допускается.

Криптографические вычисления

Для генерации ключей в TLS 1.3 используется функция HMAC-based Extract-and-Expand Key Derivation Function (HKDF), описанная в RFC 5869. Она основана на принципе «extract-then-expand». На первой фазе (HKDF-Extract) из входного ключевого материала создается псевдослучайный ключ фиксированной длины K. На второй фазе (HKDF-Expand) из ключа K получаются несколько дополнительных псевдослучайных ключей. Количество и длина выходных ключей зависят от используемого криптографического алгоритма.

Входными данными для функции HKDF-Extract являются аргументы *Salt* и *Input Keying Material* (IKM). На выходе получается псевдослучайный ключ *PRK*:

$\text{PRK} = \text{HMAC-Hash}(\text{salt}, \text{IKM})$.

В TLS 1.3 аргумент *Salt* — текущее состояние секрета, IKM — два входных секрета: ключ PSK или разделяемый секрет (EC)DHE.

9. Протоколы уровня приложений

Полный процесс получения ключей показан на рис. 9.32. Он представляет собой ступенчатый переход от секрета к секрету. Этапы идут сверху вниз, начиная от значения обозначенного «0» (строка нулевых байтов заданной длины). На схеме аргумент *Salt* для функции HKDF-Extract находится свер-

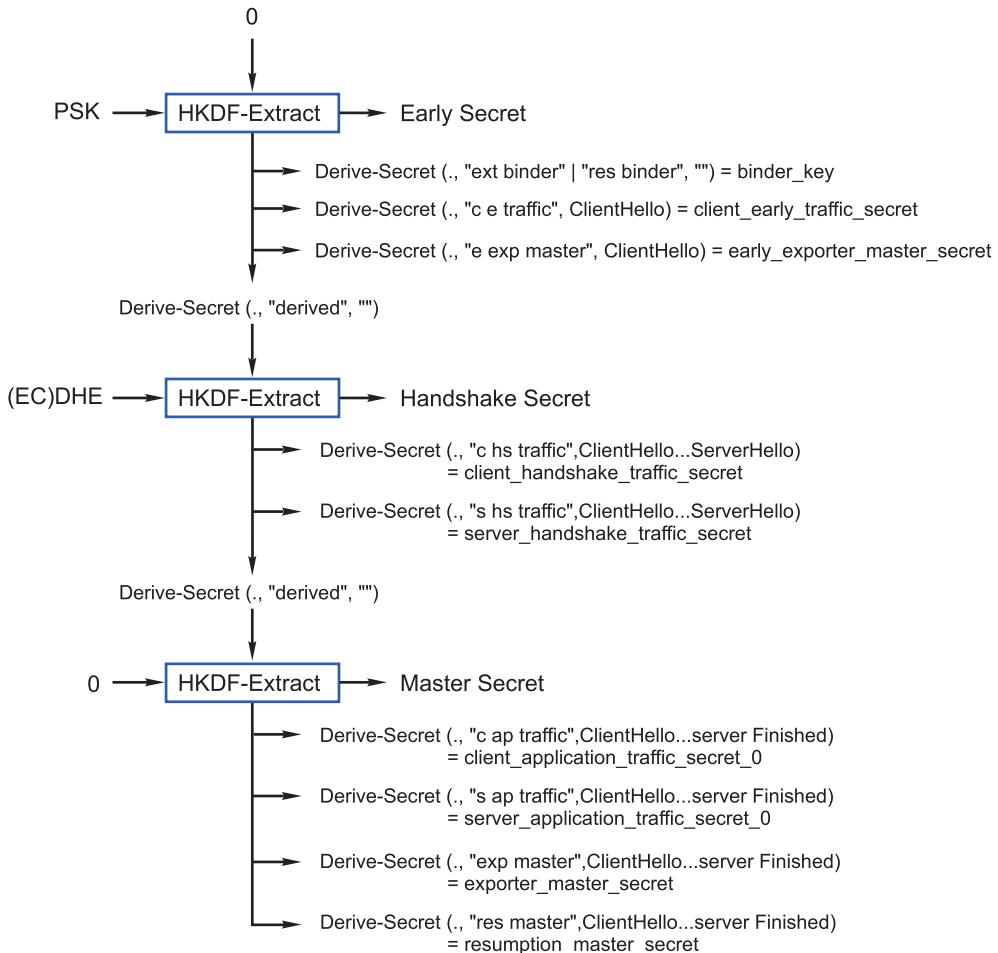


Рис. 9.32. Схема генерации ключей в TLS 1.3

ху, аргумент *IKM* — слева, выходные ключи на соответствующем этапе — справа. Например, на втором этапе результатом выполнения функции HKDF-Extract является ключ Handshake Secret, который будет использоваться для генерации промежуточных общих секретов. Для его генерации использовался разделяемый секрет (EC)DHE.

Промежуточные общие секреты получаются с помощью функции `Derive-Secret`:

```
Derive-Secret(Secret, Label, Messages) =  
HKDF-Expand-Label(Secret, Label, Transcript-  
Hash(Messages), Hash.length).
```

Функция возвращает нужное количество битов ключа — их длина передается в аргументе `Hash.length`. Аргумент `Secret` — это секрет, соответствующий текущему этапу протокола. Например, Handshake Secret является секретом для генерации ключа `server_handshake_traffic_secret`. Аргумент `Transcript-Hash` — это используемый алгоритм хеширования. Аргумент `Messages` — конкатенация указанных сообщений протокола Handshake, включая их тип и длину полей, но без заголовков протокола записи. Аргумент `Label` — это строка символов, добавляемая на соответствующем этапе. Например, «с hs traffic».

Протокол записи требует, чтобы ключи для рабочего состояния соединения генерировались из параметров безопасности, предоставленных протоколом рукопожатия.

TLS 1.3 использует алгоритмы шифрования, смоделированные в соответствии с Authenticated Encryption with Associated Data (AEAD) (RFC 5116). В качестве входных данных алгоритмы AEAD берут единственный ключ, значение `nonce`, открытый текст и дополнительную информацию.

Ключ — это `client_write_key` или `server_write_key`; `nonce` получается из порядкового номера и вектора инициализации `client_write_iv` или `server_write_iv`; дополнительная информация — заголовок записи.

Ключи, используемые протоколом записи для шифрования данных, получаются следующим образом:

```
[sender]_write_key = HKDF-Expand-Label(Secret, "key",  
"", key_length)
```

Вектор инициализации получается:

```
[sender]_write_iv = HKDF-Expand-Label(Secret, "key", "",  
iv_length)
```

- `Sender` — это сторона, отправившая сообщение (клиент или сервер);
- «`key`»/«`key`» — текстовые строки, обозначающие предназначение ключевой информации. Строки вводятся для того, чтобы результаты дополнительно различались, а разработчики, реализующие протокол, имели меньше шансов на ошибку;

- `key_length` — длина ключа;
- `iv_length` — длина вектора инициализации;
- `Secret` — секрет, соответствующий текущему этапу протокола. Для режима 0-RTT секретом является ключ `client_early_traffic_secret`; для протокола Handshake — секретом являются ключи `handshake_traffic_secret` клиента и сервера; для данных приложения — ключи `application_traffic_secret_N` клиента и сервера.

9.3.8. Протокол Record

Протокол записи SSL/TLS отвечает за идентификацию сообщений различных типов (сообщений протоколов Handshake, Change Cipher Spec и Alert, данных приложений), а также обеспечивает конфиденциальность и целостность каждого сообщения.

Последовательность работы протокола записи по доставке сообщения следующая:

- Протокол записи принимает сообщение от вышележащего уровня;
- Полученные данные разбиваются на блоки размером 2^{14} байт (16 384 байт) или меньше;
- Данные могут быть сжаты (необязательный шаг);
- Над данными вычисляется код аутентификации сообщений (MAC) и добавляется к каждой записи;
- Данные внутри каждой записи, включая MAC, шифруются с помощью согласованного протокола шифрования;
- К каждой записи добавляется заголовок.

Как только все перечисленные шаги выполнены, запись передается вниз на транспортный уровень. При получении данных выполняются аналогичные шаги, но в обратном порядке: для расшифровывания записи используется согласованный алгоритм шифрования, проверяется MAC, данные извлекаются, собираются и доставляются на вышележащий уровень (соответствующему приложению или протоколам Handshake, Change Cipher Spec, Alert).

Заголовок записи передается в открытом виде (рис. 9.33). Он содержит следующие поля: *тип содержимого* (Content type), *версия* (Version), *длина* (Length).

В TLS 1.2 поле *Version* протокола записи содержит значение 1.2 (0x0303). В TLS 1.3 значение поля *Version* зафиксировано и должно быть равно 1.2 (0x0303) во всех записях, кроме записи, содержащей первоначальное сообщение Client Hello. В нем значение поля должно иметь значение 1.0 (0x0301). Это сделано для целей совместимости. Как уже было упомянуто ранее, в TLS 1.3 информация о поддерживаемых версиях передается в расширении *supported_versions* сообщений Hello. Остальные поля с информацией о версии протокола клиентом и сервером игнорируются.

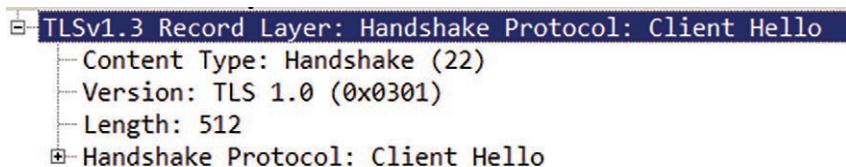


Рис. 9.33. Заголовок записи TLS 1.3

9.4. Протокол DHCP

Dynamic Host Configuration Protocol (DHCP, англ. — протокол динамической конфигурации узла) — это сетевой протокол уровня приложений, позволяющий сетевым узлам автоматически получать IP-адреса и другие сетевые параметры, необходимые для работы в IP-сети. Данный протокол работает по принципу «клиент-сервер». DHCP определен в RFC 2131 и RFC 2132 как стандарт, основанный на протоколе Bootstrap Protocol (BOOTP). DHCP позволяет узлам получать необходимую конфигурационную информацию TCP/IP от сервера DHCP. В качестве протокола транспортного уровня DHCP использует UDP. При этом применяются два номера порта: порт 67 (DHCP Server) — для отправки сообщений от клиента серверу, и порт 68 (DHCP Client) — для отправки сообщений от сервера клиенту.

Использование DHCP в средних и крупных сетях обеспечивает следующие преимущества:

- Надежная конфигурация IP-адреса. DHCP позволяет избежать ручной настройки клиентских устройств и уменьшает количество ошибок при адресации, таких как опечатки или конфликты адресов, вызванные назначением одинакового IP-адреса нескольким устройствам;
- Автоматизация и централизация сетевого управления. DHCP обеспечивает:
 - механизмы централизованной (из одного места) и автоматизированной конфигурации параметров TCP/IP;
 - возможность назначения клиентам полного набора дополнительных конфигурационных параметров с помощью опций DHCP;
 - эффективное управление изменением IP-адресов для клиентов;
 - передачу первоначальных сообщений DHCP с помощью relay-агентов, что исключает необходимость наличия DHCP-сервера в каждой подсети.

Протокол DHCP предоставляет сервис двух типов.

1. Долговременное хранение сетевых параметров для каждого клиента.
2. Выделение временных или постоянных IP-адресов для клиентов.

Основной механизм динамического выделения сетевых адресов достаточно прост: клиент запрашивает использование адреса на какой-то период времени. Сервер DHCP обязуется не назначать этот адрес другому клиенту в течение прошедшего времени и стараться предоставлять этот адрес клиенту каждый раз, когда он присыпает запрос. Период времени, на который сетевой адрес назначается клиенту, называется «арендой» (lease). Клиент может расширить время аренды, отправив последовательность запросов. Клиент может возвратить адрес серверу, если он ему больше не нужен. Также клиент может попросить о постоянном назначении адреса на неопределенное время. Даже если назначен постоянный адрес, сервер может выбрать длительное, но не бесконечное время аренды, чтобы можно было установить факт удаления клиента из сети.

9.4.1. Архитектура DHCP

Основными компонентами DHCP (рис. 9.34) являются:

- **Клиент DHCP** (DHCP client) — узел, использующий DHCP для получения конфигурационных параметров;
- **Сервер DHCP** (DHCP server) — узел, назначающий конфигурационные параметры клиентам DHCP;
- **Relay-агент DHCP** (DHCP Relay agent) — узел (маршрутизатор, коммутатор), который передает сообщения DHCP между клиентами и серверами DHCP, когда они находятся в разных подсетях.

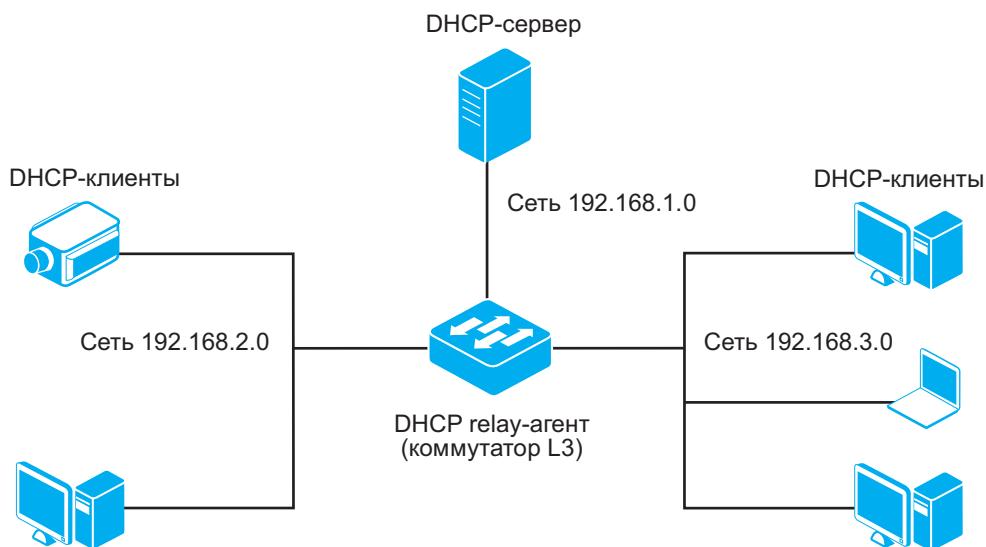


Рис. 9.34. Архитектура DHCP

Традиционный *DHCP-сервер* состоит из программного обеспечения DHCP-сервера, запущенного на серверной аппаратной платформе. Обычно DHCP-сервер не является выделенным компьютером, за исключением очень больших сетей. В большинстве случаев аппаратный сервер предоставляет сервисы DHCP наряду с выполнением других функций. Сервер не обязательно должен быть компьютером. В настоящее время маршрутизаторы и коммутаторы L3 поддерживают функциональность серверов DHCP (рис. 9.35).

Внимание: в маршрутизаторах и коммутаторах D-Link реализована поддержка функций DHCP-сервера и relay-агента. Информация о поддержке функций той или иной моделью указана в технических описаниях устройств.

DHCP-сервер выполняет две важные функции.

- Управление IP-адресами (рис. 9.36). Сервер управляет диапазоном IP-адресов и назначает их клиентам или на постоянной основе, или на определенный период времени. Он хранит таблицу привязки IP-адресов к клиентам и гарантирует, что выделенные адреса не используются несколькими клиентами.

- Предоставление сетевой конфигурации клиентам. Сервер назначает IP-адреса и предоставляет иную конфигурационную информацию, такую как имя узла, маску подсети, IP-адрес шлюза по умолчанию, IP-адрес сервера DNS, доменное имя.

Также DHCP-сервер можно настроить для выполнения дополнительных функций.

- Ответ на запросы клиентов BOOTP. Сервер прослушивает широковещательные сообщения от клиентов BOOTP и предоставляет им IP-адреса и параметры загрузки. Информация должна быть вручную настроена администратором. Сервер может одновременно выполнять функции BOOTP-сервера и DHCP-сервера.

- Перенаправление запросов. Сервер перенаправляет запросы BOOTP и DHCP соответствующим серверам в других подсетях. Сервер не предоставляет сервисы BOOTP и DHCP, когда настроен как BOOTP/DHCP relay-агент.

- Поддержка загрузки через сеть для DHCP-клиентов. Сервер может предоставлять DHCP-клиентам информацию, необходимую для загрузки через сеть: IP-адрес, параметры загрузки и информацию о конфигурации сети.

Сервер DHCP поддерживает три способа выделения IP-адресов.

- **Ручное или статическое выделение:** администратор вручную назначает адреса пользовательским устройствам на сервере (создает привязку аппаратного адреса (MAC-адреса для Ethernet) клиента к определенному IP-адресу). DHCP используется только для доставки этих настроек пользователю. Отличие этого метода от ручной настройки каждого клиента лишь в том, что настройки хранятся централизованно (на сервере DHCP), и поэтому их проще изменять при необходимости.

- **Автоматическое или постоянное выделение:** сервер автоматически выбирает адреса из диапазона (пула) адресов, определенных администратором, и выдает пользователям на постоянной основе (закрепляет за каждым пользователем).

- **Динамическое выделение:** сервер автоматически выбирает адреса из диапазона (пула) адресов, определенных администратором, но выдает пользователям на некоторое время (время аренды/lease time). По истечении времени аренды IP-адрес считается свободным, и клиент обязан запросить новый (он может оказаться тем же самым). При этом клиент может отказаться от полученного адреса. Если клиент отключился от сети, его адрес может быть выдан другому устройству при условии, что время аренды на сервере так же истекло.

9. Протоколы уровня приложений

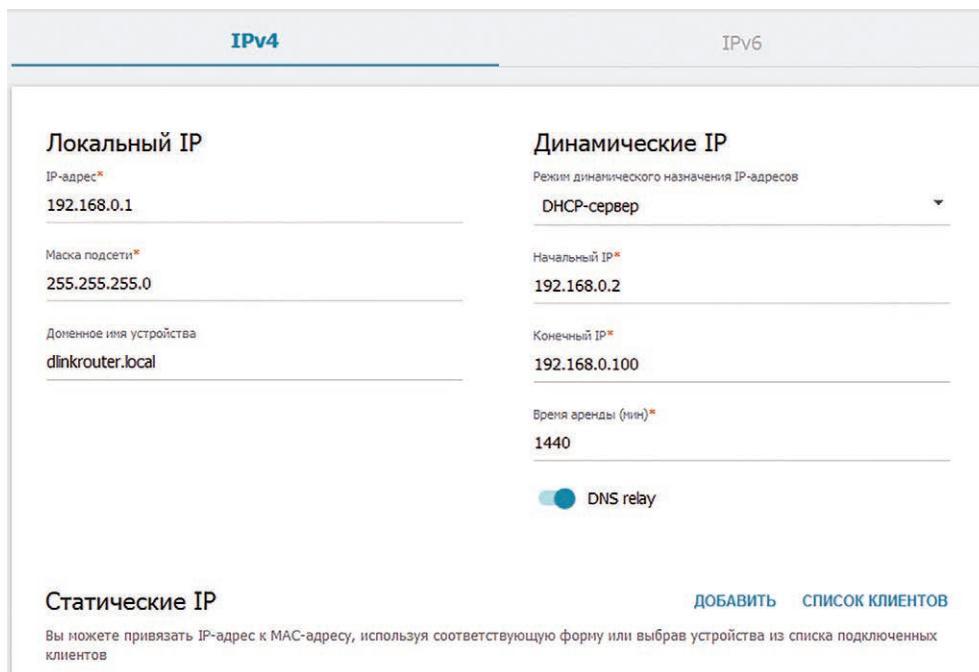


Рис. 9.35. Окно настройки DHCP-сервера на маршрутизаторе D-Link

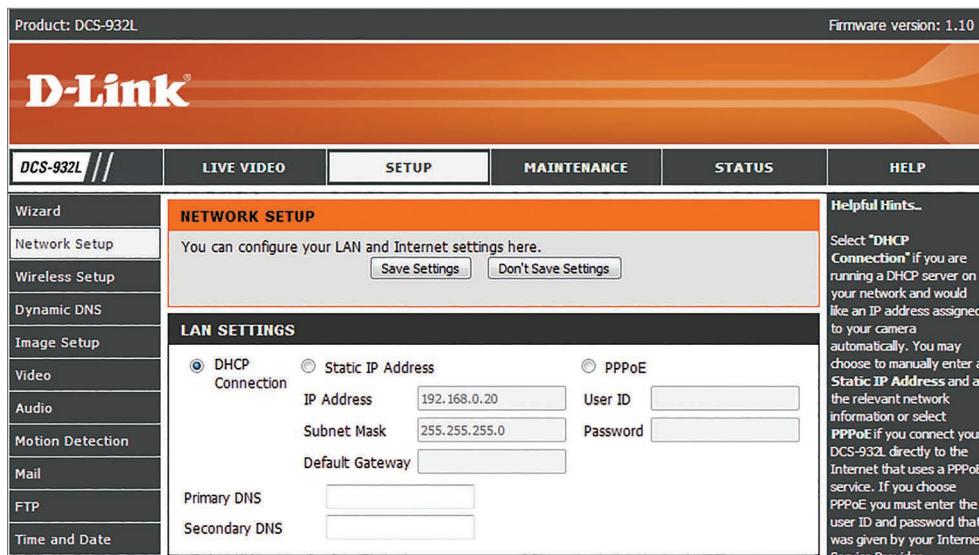


Рис. 9.36. Настройка автоматического получения параметров TCP/IP на Интернет-камере D-Link

DHCP-клиент — это программное обеспечение клиента DHCP, запущенное на клиентском устройстве. Клиентским устройством не обязательно должен быть компьютер. Им также может быть маршрутизатор, коммутатор, точка доступа, Интернет-камера (рис. 9.37) и т. д. От администратора не требуется сложной настройки DHCP-клиента. В большинстве случаев для

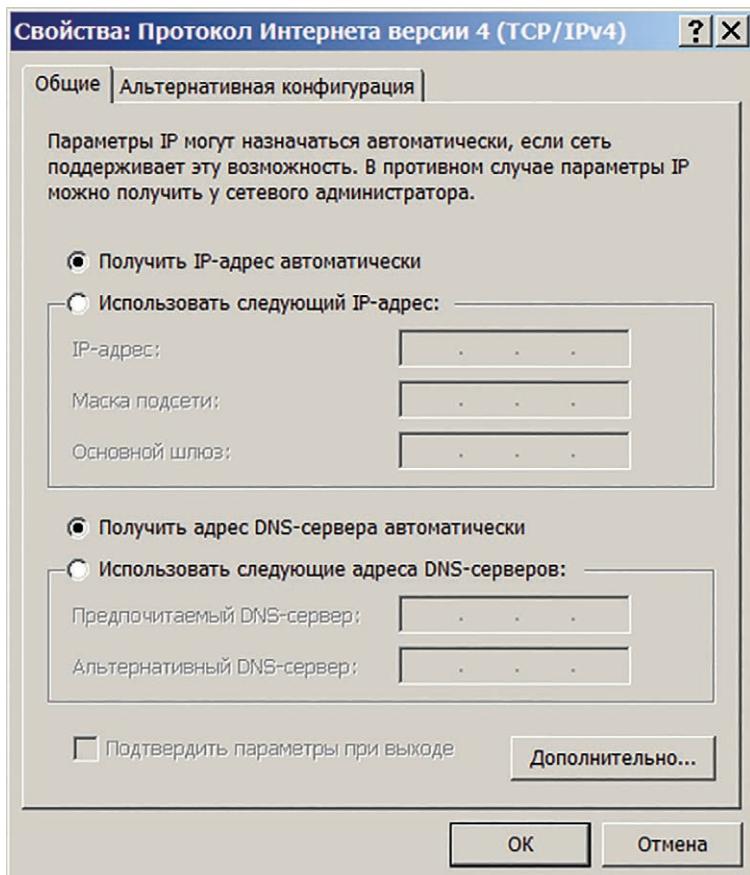


Рис. 9.37. Настройка автоматического получения параметров TCP/IP в операционной системе Windows

автоматического получения параметров TCP/IP на клиенте достаточно поставить соответствующую галочку.

На клиентском устройстве можно контролировать работу программного обеспечения DHCP-клиента. В Windows для этого используется утилита

9. Протоколы уровня приложений

ipconfig (рис. 9.38). Она позволяет посмотреть состояние текущей аренды IP-адреса, освободить IP-адрес и обновить его.

В операционной системе Linux DHCP-клиент установлен по умолчанию. Чтобы включить автоматическое получение IP-адреса на интерфейсе, требуется выполнить следующие команды.

<pre>ipconfig [/allcompartments] [/? /all /renew [адаптер] /release [адаптер] /renew6 [адаптер] /release6 [адаптер] /flushdns /displaydns /registerdns /showclassid адаптер /setclassid адаптер [идентификатор_класса]] /showclassid6 адаптер /setclassid6 адаптер [идентификатор_класса]]</pre>	
Здесь	
адаптер	Имя подключения (можно использовать знаки подстановки × and ?, см. примеры)
Параметры:	
/?	Вывод данного справочного сообщения
/all	Вывод подробных сведений о конфигурации.
/release	Освобождение адреса IPv4 для указанного адаптера.
/release6	Освобождение адреса IPv6 для указанного адаптера.
/renew	Обновление адреса IPv4 для указанного адаптера.
/renew6	Обновление адреса IPv6 для указанного адаптера.
/flushdns	Очистка кэша сопоставителя DNS.
/registerdns	Обновление всех DHCP-аренд и перерегистрация DNS-имен
/displaydns	Отображение содержимого кэша сопоставителя DNS.
/showclassid	Отображение всех допустимых для этого адаптера идентификаторов классов DHCP.
/setclassid	Изменение идентификатора класса DHCP.
/showclassid6	Отображение всех допустимых для этого адаптера идентификаторов классов DHCP IPv6.
/setclassid6	Изменение идентификатора класса DHCP IPv6.

Рис. 9.38. Параметры утилиты *ipconfig*

1. Включить интерфейс: `sudo ip link set enp1s0f0 up`
2. Запустить DHCP-клиент: `sudo dhclient enp1s0f0`
3. Посмотреть назначенный IP-адрес: `ifconfig`
`enp1s0f0` — имя сетевого интерфейса

9.4.2. Формат сообщения DHCP

Клиенты взаимодействуют с серверами с помощью сообщений DHCP для получения или обновления аренды IP-адресов. Формат сообщения DHCP определен в RFC 2131 (рис. 9.39).

В табл. 9.2 приведено описание полей сообщения DHCP.

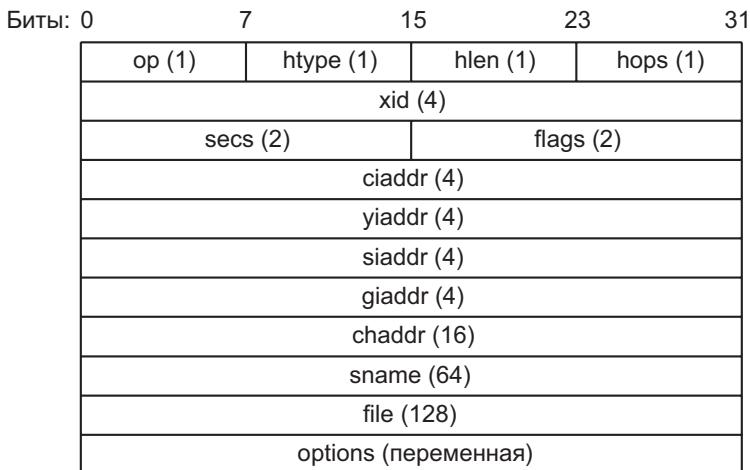


Рис. 9.39. Формат сообщения DHCP

Таблица 9.2. Описание полей сообщения DHCP

Поле	Длина, байты	Описание
<i>op</i> (Message op code/ message type)	1	Тип сообщения: 1 = BOOTREQUEST, 2 = BOOTREPLY
<i>htype</i> (Hardware address type)	1	Тип аппаратного адреса
<i>hlen</i> (Hardware address length)	1	Длина аппаратного адреса в байтах
<i>hops</i>	1	Значение устанавливается равным 0 клиентом DHCP. Опционально используется для подсчета relay-агентов, перенаправляющих сообщение
<i>xid</i> (Transaction ID)	4	Идентификатор транзакции. Произвольное число, которое используется для связи сообщений клиента и ответов на них сервера
<i>secs</i> (Seconds)	2	Количество секунд, прошедших с момента, когда клиент начал процесс получения или обновления
<i>flags</i>	2	Флаги, установленные клиентом
<i>ciaddr</i> (Client IP address)	4	Это поле заполняется только в случае, если клиент имеет IP-адрес и может отвечать на ARP-запросы

9. Протоколы уровня приложений

Окончание табл. 9.2

Поле	Длина, байты	Описание
<i>yiaddr</i> (Your IP Address)	4	IP-адрес, выданный DHCP-сервером клиенту
<i>siaddr</i> (DHCP Server IP Address)	4	IP-адрес сервера, предложившего аренду
<i>giaddr</i> (Gateway IP Address)	4	IP-адрес relay-агента DHCP
<i>chaddr</i> (Client Hardware Address)	16	Аппаратный адрес клиента
<i>sname</i> (Server Host Name)	64	Необязательное имя хоста сервера
<i>file</i> (Boot File Name)	128	Имя файла, содержащего загрузочный образ для клиента BOOTP
<i>options</i>	Переменная	Поле дополнительных параметров. Опции описаны в RFC 2132

Опции DHCP

Поле *options* сообщения DHCP состоит из одного или множества подполей — опций. Опции DHCP имеют формат, аналогичный формату BOOTP «*vendor extensions*», определенному в RFC 1497. Опции могут быть фиксированной или переменной длины. Все опции начинаются с поля *Tag* длиной 1 байт, которое уникально их идентифицирует. Далее следует поле *Length* длиной 1 байт, которое определяет количество байтов в этой опции. Оно не включает два байта, отведенных для полей *Tag* и *Length*. Третье поле *Data* переменной длины содержит отправляемые данные. Длина данных указана в поле *Length*, а тип (как их интерпретировать) — в поле *Tag*. Однако существуют два исключения. Опции 0 и 255 имеют фиксированную длину и только одно поле *Tag*. Опция 0 используется как дополнение, когда подполя необходимо выровнять по границе слова. Опция 255 используется для обозначения конца в поле специфичной для производителя информации. Другими словами, она показывает, что достигнут конец поля *options* сообщения DHCP.

Опции описаны в RFC 2132 и ряде других RFC. Некоторые из них приведены в табл. 9.3.

Таблица 9.3. Опции DHCP

Tag	Имя	Длина, байты	Описание
0	Pad	0	Дополнение до границы слова
1	Subnet Mask	4	Значение маски подсети
3	Router	Переменная (умноженная на 4)	IP-адрес маршрутизатора

Окончание табл. 9.2

Tag	Имя	Длина, байты	Описание
6	Domain Server	Переменная (умноженная на 4)	IP-адрес сервера DNS
12	Hostname	Переменная	Имя клиента
15	Domain Name	Переменная	Доменное имя клиента
33	Static Route	Переменная (умноженная на 8)	Список статических маршрутов
50	Address Request	4	Запрашиваемый IP-адрес
53	DHCP Msg Type	1	Тип сообщения DHCP
54	DHCP Server Id	4	Идентификатор DHCP-сервера
58	Renewal Time	4	Время обновления DHCP
60	Class Id	Переменная	Идентификатор класса
61	Client Id	Переменная	Идентификатор клиента
82	Relay Agent Information	Переменная	Информация relay-агента
125	V-I Vendor-Specific Information		Информация, определяемая производителем

9.4.3. Взаимодействие между клиентом и сервером DHCP

Для получения сетевых настроек клиент обменивается с сервером серией сообщений (рис. 9.40).

Рассмотрим подробно последовательность обмена сообщениями.

1. Обнаружение DHCP. Клиент начинает процесс в состоянии INIT. Он пытается обнаружить DHCP-серверы в своей локальной подсети и отправляет широковещательное сообщение DHCPDISCOVER (на канальном уровне MAC-адрес назначения FF:FF:FF:FF:FF:FF, на сетевом уровне IP-адрес назначения 255.255.255.255). Поскольку у клиента еще нет IP-адреса, то IP-адрес источника сообщения равен 0.0.0.0, MAC-адрес источника равен MAC-адресу клиента. Сообщение содержит опции, которые включают идентификационную информацию клиента и список запрашиваемых у сервера параметров (маску подсети, IP-адрес шлюза по умолчанию, IP-адрес сервера DNS и др.). Если в подсети настроен relay-агент, он может перехватить и перенаправить сообщение DHCP-серверам, находящимся в других подсетях.

Как серверы DHCP, которые получат это сообщение, смогут определить сеть, в которой находится клиент, и выдать IP-адрес из правильного диапазона, если их несколько? Для этого они анализируют следующую информацию:

- IP-адрес интерфейса, на который пришел запрос;
- Наличие IP-адреса relay-агента в запросе.

Сервер должен определить, находится ли клиент в той же сети, к которой подключен его интерфейс, или для передачи запроса использовался relay-агент. Когда используется relay-агент, в поле *giaddr* (Gateway IP Address = Relay

9. Протоколы уровня приложений

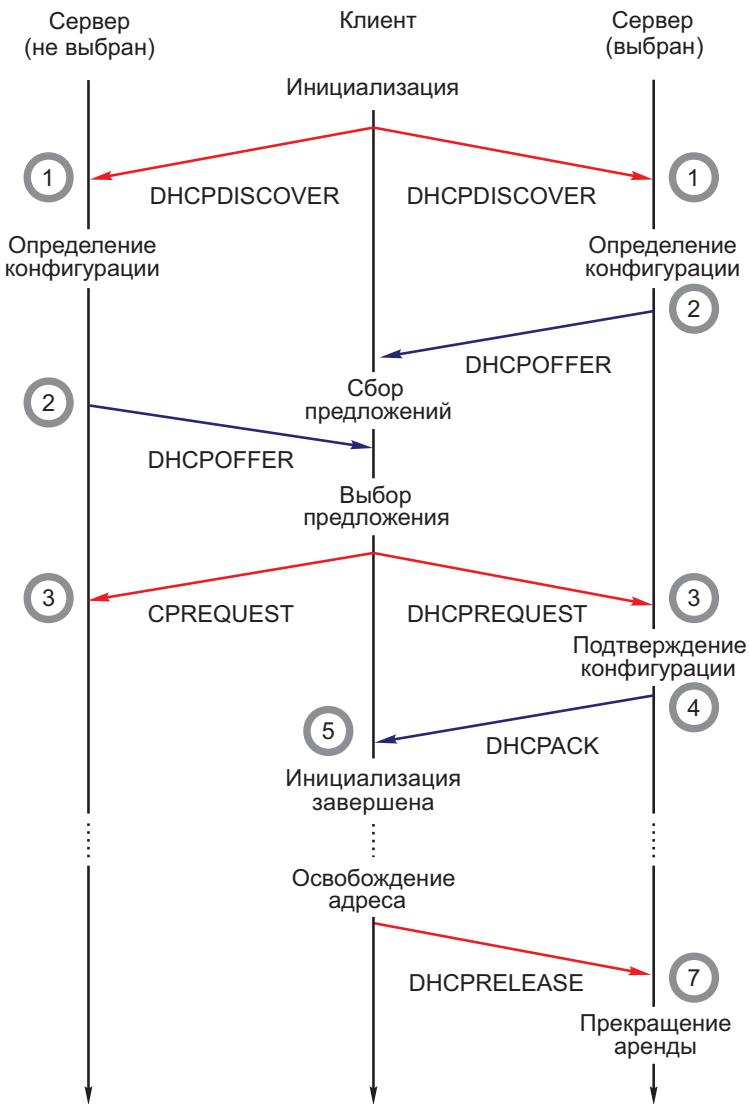


Рис. 9.40. Временная диаграмма обмена сообщениями DHCP между клиентом и сервером

agent address) сообщения указан его IP-адрес. Если агент не используется, в этом поле указано значение 0.0.0.0. Далее сервер по IP-адресу интерфейса или relay-агента ищет нужный диапазон. Если диапазон существует, из него выбирается IP-адрес.

После отправления сообщения DHCPDISCOVER клиент переходит в состояние SELECTING и ожидает ответы на него.

```
⊕ Bootstrap Protocol (Discover)
    - Message type: Boot Request (1)
    - Hardware type: Ethernet (0x01)
    - Hardware address length: 6
    - Hops: 0
    - Transaction ID: 0xcccb6abd9
    - Seconds elapsed: 0
    ⊕ Bootp flags: 0x0000 (Unicast)
    ⊕ Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: 00:33:22:33:44:55 (00:33:22:33:44:55)
    Client hardware address padding: 000000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    ⊕ Option: (53) DHCP Message Type (Discover)
    ⊕ Option: (61) Client identifier
    ⊕ Option: (12) Host Name
    ⊕ Option: (60) Vendor class identifier
    ⊕ Option: (55) Parameter Request List
        - Length: 12
        - Parameter Request List Item: (1) Subnet Mask
        - Parameter Request List Item: (15) Domain Name
        - Parameter Request List Item: (3) Router
        - Parameter Request List Item: (6) Domain Name Server
        - Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
        - Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
        - Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
        - Parameter Request List Item: (31) Perform Router Discover
        - Parameter Request List Item: (33) Static Route
        - Parameter Request List Item: (121) Classless Static Route
        - Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
        - Parameter Request List Item: (43) Vendor-Specific Information
    ⊕ Option: (255) End
```

Рис. 9.41. Сообщение DHCPDISCOVER

2. Предложение DHCP. Получив сообщение от клиента, каждый сервер определяет требуемую конфигурацию в соответствии с указанными сетевым администратором настройками. При выделении нового адреса серверы должны проверить, что этот адрес в настоящий момент не используется. После этого они отправляют в ответ клиенту сообщение DHCPOFFER (рис. 9.42). Оно содержит предлагаемый IP-адрес в поле *Your IP Address* и другие конфигурационные параметры в поле *options* (маску подсети, адрес шлюза по умолчанию, адрес DNS-сервера, время аренды и др.). Каждый сервер резервирует предлагаемый адрес до тех пор, пока клиент не примет предложение. Если клиент находится в другой сети, сервер передает сообщение DHCPOFFER через relay-агента.

Сообщение DHCPOFFER может отправляться широковещательно на IP-адрес 255.255.255.255, на IP-адрес, выделенный клиенту, или на IP-адрес relay-агента, если он используется. Может возникнуть вопрос, как сообщение дойдет до клиента, который еще не получил IP-адрес? В сообщениях DHCPDISCOVER и DHCPOFFER поле *chaddr* (Client MAC address) содержит MAC-адрес клиента. Когда сервер или relay-агент находится с клиентом в пределах одного

```

Bootstrap Protocol (Offer)
  - Message type: Boot Reply (2)
  - Hardware type: Ethernet (0x01)
  - Hardware address length: 6
  - Hops: 0
  - Transaction ID: 0xcccb6abd9
  - Seconds elapsed: 0
  + Bootp flags: 0x0000 (Unicast)
  - Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.0.10
  - Next server IP address: 192.168.0.1
  - Relay agent IP address: 0.0.0.0
    Client MAC address: 00:33:22:33:44:55 (00:33:22:33:44:55)
  - Client hardware address padding: 000000000000000000000000
  - Server host name not given
  - Boot file name not given
  - Magic cookie: DHCP
  + Option: (53) DHCP Message Type (Offer)
  + Option: (54) DHCP Server Identifier
  + Option: (51) IP Address Lease Time
    Length: 4
    IP Address Lease Time: (86400s) 1 day
  + Option: (58) Renewal Time Value
  + Option: (59) Rebinding Time Value
  + Option: (1) Subnet Mask
    Length: 4
    Subnet Mask: 255.255.255.0
  + Option: (28) Broadcast Address
  + Option: (6) Domain Name Server
    Length: 4
    Domain Name Server: 192.168.0.1
  + Option: (3) Router
    Length: 4
    Router: 192.168.0.1
  + Option: (15) Domain Name
  + Option: (255) End

```

Рис. 9.42. Сообщение DHCPOFFER

широковещательного домена (подключены к одному сегменту сети), они указывают в качестве MAC-адреса назначения сообщения MAC-адрес клиента.

Сообщение DHCPDISCOVER содержит номер транзакции Transaction ID (поле *xid*). Сервер в своем предложении DHCPOFFER должен использовать номер транзакции из сообщения клиента. В противном случае клиент отбросит ответ от сервера, поскольку посчитает, что это ответ другой транзакции.

3. Запрос DHCP. Клиент получает одно или несколько сообщений DHCPOFFER от одного или нескольких DHCP-серверов. Если предложений несколько, он выбирает лучшее из них и широковещательно отправляет выбранному серверу сообщение DHCPREQUEST (рис. 9.43) с целью запроса предложенных конфигурационных параметров. Сообщение DHCPREQUEST

```

Bootstrap Protocol (Request)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xccb6abd9
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: 00:33:22:33:44:55 (00:33:22:33:44:55)
  Client hardware address padding: 000000000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Request)
  Option: (61) Client identifier
    Option: (50) Requested IP Address
      Length: 4
      Requested IP Address: 192.168.0.10
    Option: (54) DHCP Server Identifier
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  Option: (12) Host Name
  Option: (81) Client Fully Qualified Domain Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (255) End

```

Рис. 9.43. Сообщение DHCPREQUEST

должно включать опцию *Server identifier*, которая содержит IP-адрес выбранного сервера и опцию *Requested IP address*, содержащую предложенный IP-адрес. Широковещательная отправка сообщения гарантирует, что все серверы, отправившие предложения, будут проинформированы о выборе клиента. Если необходимо, это сообщение будет перенаправлено relay-агентом соответствующим серверам. Серверы, которые не были выбраны клиентом, могут отменить резервирование предложенного IP-адреса.

После отправки сообщения DHCPREQUEST клиент переходит в состояние REQUESTING и ожидает ответ от выбранного сервера.

4. Подтверждение DHCP. Сервер, выбранный клиентом, сохраняет конфигурационные параметры клиента в постоянной памяти. Он также отправляет клиенту одноадресное подтверждение DHCPOFFER (рис. 9.44), содержащее конфигурационные параметры, аналогичные ранее предложенным в сообщении DHCPOFFER.

Рис. 9.44. Сообщение DHCPACK

Комбинация MAC-адреса клиента и назначенного ему IP-адреса становится уникальным идентификатором времени аренды.

Когда клиент получает подтверждение DHCPACK, он должен выполнить финальную проверку конфигурационных параметров и установить два таймера — T1 и T2. Проверка уникальности IP-адреса в пределах широковещательного домена выполняется путем отправки ARP-запроса на выделенный IP-адрес. Если ответ не получен, значит, адрес в пределах данной подсети уникален. Клиент настраивает сетевой интерфейс в соответствии с полученными параметрами и переходит в состояние BOUND.

Если обнаружится, что адрес уже используется, клиент должен отправить серверу сообщение DHCPDECLINE и перезапустить процесс получения конфигурационных параметров, т. е. отправить сообщение DHCPDISCOVER через определенное время.

```
Bootstrap Protocol (Release)
  - Message type: Boot Request (1)
  - Hardware type: Ethernet (0x01)
  - Hardware address length: 6
  - Hops: 0
  - Transaction ID: 0x1d00c707
  + Seconds elapsed: 3
  + Bootp flags: 0x0000 (Unicast)
  - Client IP address: 192.168.0.10
  - Your (client) IP address: 0.0.0.0
  - Next server IP address: 0.0.0.0
  - Relay agent IP address: 0.0.0.0
  - Client MAC address: 00:33:22:33:44:55 (00:33:22:33:44:55)
  - Client hardware address padding: 00000000000000000000000000000000
  - Server host name not given
  - Boot file name not given
  - Magic cookie: DHCP
  + Option: (53) DHCP Message Type (Release)
  + Option: (54) DHCP Server Identifier
    - Length: 4
    - DHCP Server Identifier: 192.168.0.1
  + Option: (61) Client identifier
    - Length: 7
    - Hardware type: Ethernet (0x01)
    - Client MAC address: 00:33:22:33:44:55 (00:33:22:33:44:55)
  + Option: (255) End
  - Padding: 000000000000000000000000000000000000000000000000000000000000000...
```

Рис. 9.45. Сообщение DHCPRELEASE

Если выбранный сервер не может удовлетворить запрос DHCPREQUEST (например, запрашиваемый адрес уже выделен), он отправляет клиенту отрицательное подтверждение DHCPNAK. После его получения клиент перезапускает процесс получения конфигурационных параметров.

Клиент может отказаться от аренды IP-адреса (если он ему больше не нужен), отправив серверу уведомительное сообщение DHCPRELEASE (рис. 9.45).

Представьте ситуацию, что клиент по какой-то причине перезагрузился, но выделенное время аренды адреса еще не истекло (аренду можно получить в диапазоне от нескольких минут до лет). В этом случае клиенту не требуется проходить весь описанный выше процесс. Он может опустить некоторые шаги.

Алгоритм повторного использования адреса следующий.

1. Клиент начинает процесс повторного использования адреса в состоянии INIT-REBOOT, а не в INIT. Он отправляет в свою локальную подсеть широковещательный запрос DHCPREQUEST (рис. 9.46). IP-адрес источника в этом сообщении равен 0.0.0.0. В опции *Requested IP address* (запрашиваемый IP-адрес) сообщения клиент указывает назначенный ему ранее IP-адрес. В опции *Parameter Request List* могут быть указаны дополнительные параметры.

9. Протоколы уровня приложений

```
Ethernet II, Src: WistronI_72:a5:82 (20:6a:8a:72:a5:82), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Request)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xba820b55
  Seconds elapsed: 0
  Boot flags: 0x8000, Broadcast flag (Broadcast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Request)
  Option: (61) Client identifier
  Option: (50) Requested IP Address
    Length: 4
    Requested IP Address: 192.168.0.3
  Option: (12) Host Name
  Option: (81) Client Fully Qualified Domain Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (255) End
  Padding: 00000000
```

Рис. 9.46. Сообщение DHCPREQUEST, отправляемое при повторном использовании адреса

Опция *Server identifier* должна отсутствовать. Для передачи сообщения серверам, находящимся в другой подсети, используется relay-агент.

2. Сервер, который знает конфигурационные параметры клиента, посылает ему в ответ сообщение DHCPCACK. После его получения клиент проверяет уникальность адреса и, если все в порядке, настраивает на интерфейсе сетевые параметры.

Если запрос клиента некорректный или адрес уже используется, сервер отправит ему отрицательное подтверждение DHCPNAK. Если адрес не уникален, клиент отправит серверу сообщение DHCPDECLINE. Сообщения DHCPNAK и DHCPDECLINE потребуют от клиента перезапуска процесса получения конфигурационных параметров.

Клиент хранит два таймера — T1 и T2. Таймер T1 показывает время, через которое клиент попытается продлить у сервера аренду выданного ранее IP-адреса. Если от него не будет получен ответ, то по истечении таймера T2 клиент попытается продлить аренду адреса у любого сервера. Длительность T1 должна быть меньше, чем T2.

По умолчанию таймер T1 устанавливается равным половине времени аренды. Когда определенное им время истекает, клиент переходит из состояния

Технологии TCP/IP в современных компьютерных сетях

```
Frame 3327: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Ethernet II, Src: D-LinkIn_9e:74:f0 (74:da:da:9e:74:f0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 67, Dst Port: 68
Bootstrap Protocol (NAK)
  Message type: Boot Reply (2)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xed766995
  Seconds elapsed: 0
  Boot flags: 0x8000, Broadcast flag (Broadcast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: Giga-Byt_5e:d3:87 (fc:aa:14:5e:d3:87)
  Client hardware address padding: 00000000000000000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (NAK)
  Option: (54) DHCP Server Identifier
    Length: 4
    DHCP Server Identifier: 192.168.0.1
  Option: (56) Message
    Length: 13
    Message: wrong network
  Option: (255) End
  Padding: 000000000000000000000000000000000000000000000000000000000000000...
```

Рис. 9.47. Сообщение DHCPNAK

BOUND в состояние обновления (RENEWING). Следует отметить, что клиент может инициировать обновление времени аренды до истечения таймера T1.

Алгоритм обновления времени аренды следующий:

1. Клиент создает и отправляет на уникальный адрес сервера сообщение DHCPREQUEST. Оно содержит IP-адрес клиента в поле *Client IP address* и время аренды. Опция *Server identifier* должна в сообщении отсутствовать. Также клиент запоминает локальное время отправки сообщения, которое будет использоваться для вычисления времени окончания аренды.

2. Предположим, что сервер доступен и что он получает сообщение от клиента. Если сервер согласен продлить аренду, он отправляет клиенту сообщение DHCPACK. В противном случае отправляется сообщение DHCPNAK (рис. 9.47).

3. Когда клиент получает от сервера сообщение DHCPACK, он вычисляет время окончания аренды, обновляет таймеры T1 и T2 и переходит в состояние BOUND.

4. Если клиент не получил ответ от сервера, он останется в состоянии RENEWING и будет регулярно отправлять ему сообщения DHCPREQUEST. По истечении времени, определенного таймером T2 (оно равно 87,5 % времени аренды), клиент переходит в состояние REBINDING.

5. Не получив ответ от сервера, первоначально выдавшего адрес в аренду, клиент отправляет в свою локальную подсеть широковещательный запрос DHCPREQUEST. Он содержит IP-адрес клиента в поле *Client IP address*.

6. Сервер, который знает время аренды клиента и соглашается ее продлить, посыпает ему в ответ сообщение DHCPACK. Если сервер не знает время аренды или не соглашается на продление, он отправляет сообщение DHCPNAK.

7. Когда клиент получает от сервера сообщение DHCPACK, он вычисляет время окончания аренды, обновляет таймеры T1 и T2 и переходит в состояние BOUND.

8. Когда клиент получает от сервера сообщение DHCPNAK, он понимает, что аренду продлить нельзя и переходит в состояние INIT.

9. Если время аренды истекло, но клиент не получил сообщение DHCPACK, он переходит в состояние INIT и пытается получить в аренду новый адрес.

9.4.4. Функционирование relay-агента DHCP

Сообщения DHCP передаются широковещательно. Чтобы клиент и сервер могли ими обмениваться, они должны находиться в одном широковещательном домене (в одной IP-сети или подсети). Это связано с тем, что маршрутизаторы не передают широковещательные пакеты на другие интерфейсы. Таким образом, если клиент и сервер DHCP находятся в разных сетях, для передачи сообщений между ними нужен посредник. Им является relay-агент DHCP.

Relay-агент DHCP — это любой узел, маршрутизатор или коммутатор, который настроен для передачи пакетов DHCP между клиентом и сервером, находящихся в разных сетях. Передача сообщений DHCP relay-агентом (рис. 9.48) отличается от передачи IP-пакетов маршрутизатором.

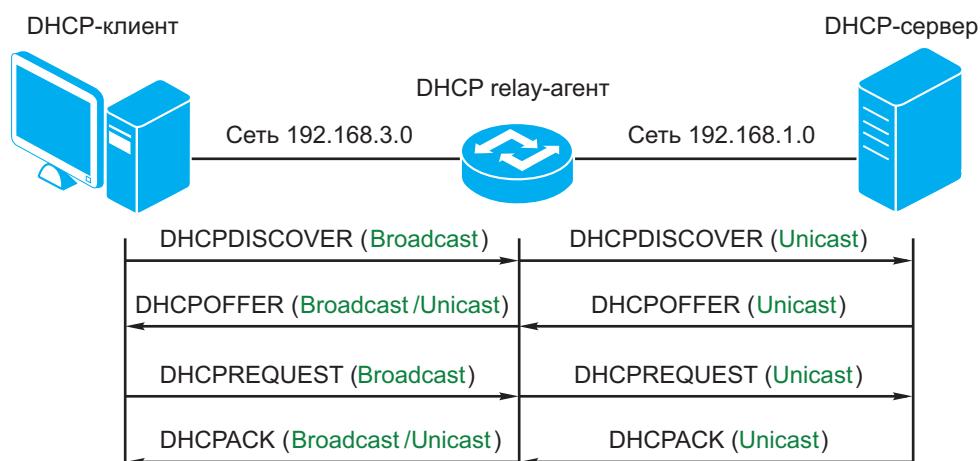


Рис. 9.48. Передача сообщений через DHCP relay-агента

Маршрутизатор при передаче IP-пакета не изменяет в нем IP-адреса приемника и назначения. Relay-агент, получив сообщение DHCP, генерирует новое сообщение и отправляет его через соответствующий интерфейс.

Давайте рассмотрим процедуру получения параметров TCP/IP в сети, где настроен DHCP relay-агент.

1. Клиент широковещательно отправляет сообщение DHCPDISCOVER (рис. 9.49) с целью обнаружения DHCP-серверов в своей сети. Если в его сети DHCP-серверы отсутствуют, этот пакет получит relay-агент. Relay-агент знает, в какие сети передавать DHCP-сообщение, так как в его настройках указаны IP-адреса DHCP-серверов. Прежде чем передать сообщение DHCPDISCOVER, relay-агент выполняет в нем следующие изменения:

- В поле *Destination MAC address* широковещательный MAC-адрес, указанный клиентом, заменяется MAC-адресом сервера.
 - В поле *Source MAC address* MAC-адрес клиента заменяется MAC-адресом интерфейса relay-агента.
 - В поле *Destination IP address* широковещательный IP-адрес назначения, указанный клиентом, заменяется IP-адресом DHCP-сервера.
 - В поле *Source IP address* IP-адрес 0.0.0.0 заменяется IP-адресом интерфейса relay-агента.
 - В опции *Relay Agent IP address* значение 0.0.0.0 заменяется IP-адресом интерфейса relay-агента, через который сообщение DHCPDISCOVER было получено.

Рис. 9.49. Сообщение DHCPDISCOVER, отправленное клиентом

9. Протоколы уровня приложений

Рис. 9.50. Сообщение DHCPDISCOVER, перенаправленное relay-агентом DHCP-серверу

Рис. 9.51. Сообщение DHCPOFFER, отправленное DHCP-сервером relay-агенту

После всех изменений relay-агент отправляет серверу одноадресное сообщение DHCPDISCOVER (рис. 9.50).

2. DHCP-сервер, проанализировав сообщение DHCPDISCOVER, обнаруживает, что оно отправлено через relay-агента, поскольку в поле *Relay Agent IP address* указан его IP-адрес. Используя IP-адрес relay-агента, DHCP-сервер ищет соответствующий пул адресов и выбирает из него свободный адрес для клиента. Далее сервер отправляет клиенту сообщение DHCPOFFER (рис. 9.51), в котором в качестве MAC- и IP-адреса назначения указаны MAC- и IP-адрес relay-агента, от которого было получено сообщение DHCPDISCOVER.

Прежде чем передать полученное сообщение клиенту, relay-агент выполняет в нем следующие изменения:

- В поле *Destination MAC address* MAC-адрес relay-агента, заменяется:
 - широковещательным MAC-адресом (FF:FF:FF:FF:FF:FF), если флаг Broadcast в сообщении DHCPOFFER, полученном от сервера, равен 1;
 - MAC-адресом клиента (указан в поле *Client MAC address*), если флаг Broadcast равен 0.
 - В поле *Source MAC address* MAC-адрес сервера заменяется MAC-адресом интерфейса relay-агента.
 - В поле *Destination IP address* IP-адрес relay-агента заменяется:

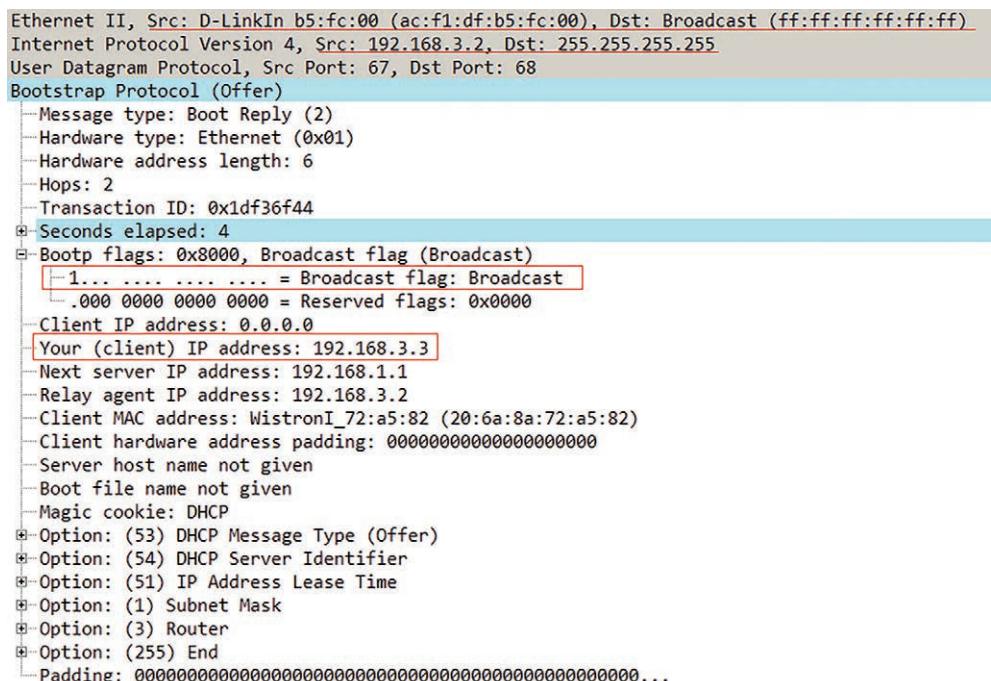


Рис. 9.52. Сообщение DHCPOFFER, перенаправленное relay-агентом клиенту

9. Протоколы уровня приложений

а) широковещательным IP-адресом (255.255.255.255), если флаг Broadcast в сообщении DHCPOFFER (рис. 9.52), полученном от сервера, равен 1;

б) выделенным клиенту IP-адресом (указан в поле *Your (client) IP address*), если флаг Broadcast равен 0.

— В поле *Source IP address* IP-адрес сервера заменяется IP-адресом интерфейса relay-агента.

3. DHCP-клиент, получивший предложение, широковещательно отправляет сообщение DHCPREQUEST (рис. 9.53) с целью запроса предложенных конфигурационных параметров у выбранного сервера. Это сообщение получает relay-агент, который изменяет в нем адресную информацию в заголовках сетевого и канального уровня, а также добавляет свой IP-адрес в поле *Relay Agent IP address* (аналогично изменениям в сообщении DHCPDISCOVER). После всех изменений relay-агент отправляет одноадресное сообщение DHCPREQUEST (рис. 9.54) нужному серверу.

4. Сервер, выбранный клиентом, отправляет ему одноадресное подтверждение DHCPACK (рис. 9.55), содержащее конфигурационные параметры, аналогичные ранее предложенным в сообщении DHCPOFFER. В сообщении DHCPACK в качестве MAC- и IP-адреса назначения указаны MAC- и IP-адрес соответствующего relay-агента. Прежде чем передать полученное

```
Ethernet II, Src: WistronI_72:a5:82 (20:6a:8a:72:a5:82), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Request)

Message type: Boot Request (1)
Hardware type: Ethernet (0x01)
Hardware address length: 6
Hops: 0
Transaction ID: 0x1df36f44
Seconds elapsed: 4
Bootp flags: 0x8000, Broadcast flag (Broadcast)
Client IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
Client hardware address padding: 000000000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type (Request)
Option: (61) Client identifier
Option: (50) Requested IP Address
Option: (54) DHCP Server Identifier
Option: (12) Host Name
Option: (81) Client Fully Qualified Domain Name
Option: (60) Vendor class identifier
Option: (55) Parameter Request List
Option: (255) End
```

Рис. 9.53. Сообщение DHCPREQUEST, отправленное клиентом

Технологии TCP/IP в современных компьютерных сетях

```
Ethernet II, Src: D-LinkIn b5:fc:00 (ac:f1:df:b5:fc:00), Dst: D-LinkIn 1f:9c:01 (84:c9:b2:1f:9c:01)
Internet Protocol Version 4, Src: 192.168.3.2, Dst: 192.168.1.1
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Request)
- Message type: Boot Request (1)
- Hardware type: Ethernet (0x01)
- Hardware address length: 6
- Hops: 1
- Transaction ID: 0x1df36f44
- Seconds elapsed: 4
- Bootp flags: 0x8000, Broadcast flag (Broadcast)
- Client IP address: 0.0.0.0
- Your (client) IP address: 0.0.0.0
- Next server IP address: 0.0.0.0
- Relay agent IP address: 192.168.3.2
- Client MAC address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
- Client hardware address padding: 000000000000000000000000
- Server host name not given
- Boot file name not given
- Magic cookie: DHCP
- Option: (53) DHCP Message Type (Request)
- Option: (61) Client identifier
- Option: (50) Requested IP Address
- Option: (54) DHCP Server Identifier
- Option: (12) Host Name
- Option: (81) Client Fully Qualified Domain Name
- Option: (60) Vendor class identifier
- Option: (55) Parameter Request List
- Option: (255) End
```

Рис. 9.54. Сообщение DHCPREQUEST, перенаправленное relay-агентом DHCP-серверу

```
Ethernet II, Src: D-LinkIn 1f:9c:01 (84:c9:b2:1f:9c:01), Dst: D-LinkIn b5:fc:00 (ac:f1:df:b5:fc:00)
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.3.2
User Datagram Protocol, Src Port: 67, Dst Port: 67
Bootstrap Protocol (ACK)
- Message type: Boot Reply (2)
- Hardware type: Ethernet (0x01)
- Hardware address length: 6
- Hops: 1
- Transaction ID: 0x1df36f44
- Seconds elapsed: 4
- Bootp flags: 0x8000, Broadcast flag (Broadcast)
  |.... .... .... = Broadcast flag: Broadcast
  |...000 0000 0000 0000 = Reserved flags: 0x0000
- Client IP address: 0.0.0.0
- Your (client) IP address: 192.168.3.3
- Next server IP address: 192.168.1.1
- Relay agent IP address: 192.168.3.2
- Client MAC address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
- Client hardware address padding: 000000000000000000000000
- Server host name not given
- Boot file name not given
- Magic cookie: DHCP
- Option: (53) DHCP Message Type (ACK)
- Option: (54) DHCP Server Identifier
- Option: (51) IP Address Lease Time
- Option: (1) Subnet Mask
- Option: (3) Router
- Option: (255) End
- Padding: 000000000000000000000000000000000000000000000000...
```

Рис. 9.55. Сообщение DHCPACK, отправленное DHCP-сервером relay-агенту

9. Протоколы уровня приложений

Рис. 9.56. Сообщение DHCPACK, перенаправленное relay-агентом клиенту

сообщение клиенту, relay-агент выполняет в DHCPOFFER (рис. 9.56) изменения, аналогичные изменениям, производимым с сообщением DHCPOFFER.

Рассмотрим пример настройки DHCP relay-агента в офисной сети (рис. 9.57). Предположим, что сеть разбита на подсети. Клиенты находятся в подсетях 192.168.2.0/24 и 192.168.3.0/24 и настроены для автоматического получения параметров TCP/IP. Чтобы исключить необходимость размещения DHCP-сервера в каждой из подсетей, решено использовать функцию DHCP Relay на коммутаторе L3 SW2. Он будет перенаправлять сообщения централизованному DHCP-серверу (рис. 9.58), расположенному в отдельной

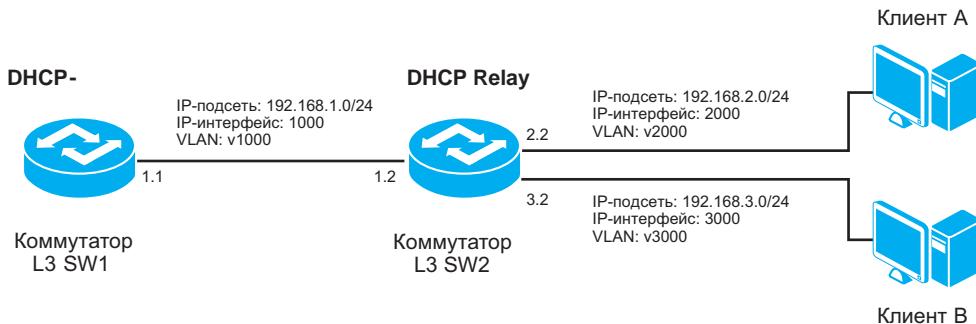


Рис. 9.57. Схема сети

подсети. DHCP-сервер, настроенный на коммутаторе L3 SW1, выдает конфигурационные параметры клиентам в подсетях 192.168.2.0/24 и 192.168.3.0/24 (рис. 9.58).

Настройка коммутатора L3 SW1 (приведена для модели DGS-3620-24TC)

Удалить порты из VLAN по умолчанию:

config vlan default delete 1–4

Создать VLAN v1000 для DHCP-сервера и настроить немаркованные порты:

create vlan v1000 tag 1000

config vlan v1000 add untagged 1–4

Создать IP-интерфейс для DHCP-сервера:

create ipif 1000 192.168.1.1/24 v1000 state enable

Создать диапазон адресов для подсети 192.168.2.0/24:

create dhcp pool pool1

config dhcp pool network_addr pool1 192.168.2.0/24

Задать шлюз по умолчанию для подсети 192.168.2.0/24:

config dhcp pool default_router pool1 192.168.2.2

Указать, какие IP-адреса не будут выдаваться клиентам:

create dhcp excluded_address begin_address 192.168.2.1 end_address 192.168.2.2

```
DGS-3620-28TC:admin#show dhcp pool
Command: show dhcp pool

Pool Name          :pool1
Network Address   :192.168.2.0/24
Domain Name        :
DNS Server         :
NetBIOS Name Server:
NetBIOS Node Type :NOT SET
Default Router    :192.168.2.2
Pool Lease         :1 Days, 0 Hours, 0 Minutes
Boot File          :
Next Server        :
Option Profile    :

Pool Name          :pool2
Network Address   :192.168.3.0/24
Domain Name        :
DNS Server         :
NetBIOS Name Server:
NetBIOS Node Type :NOT SET
Default Router    :192.168.3.2
Pool Lease         :1 Days, 0 Hours, 0 Minutes
Boot File          :
Next Server        :
Option Profile    :
```

Рис. 9.58. Созданные пулы адресов на DHCP-сервере

9. Протоколы уровня приложений

Создать диапазон адресов для подсети 192.168.3.0/24:

create dhcp pool pool2

config dhcp pool network_addr pool1 192.168.3.0/24

Задать шлюз по умолчанию для подсети 192.168.3.0/24:

config dhcp pool default_router pool2 192.168.3.2

Указать, какие IP-адреса не будут выдаваться клиентам:

create dhcp excluded_address begin_address 192.168.3.1 end_address 192.168.3.2

Запустить DHCP-сервер:

enable dhcp-server

Создать маршрут по умолчанию:

create iproute default 192.168.1.2

Настройка коммутатора L3 SW2 (приведена для модели DGS-3120-24TC)

Удалить порты из VLAN по умолчанию:

config vlan default delete 1-13

Создать VLAN v1000 и настроить немаркованные порты:

create vlan v1000 tag 1000

config vlan v1000 add untagged 1-4

Создать VLAN v2000 и настроить немаркованные порты:

create vlan v2000 tag 2000

config vlan v2000 add untagged 5-8

Создать VLAN v3000 и настроить немаркованные порты:

create vlan v3000 tag 3000

config vlan v3000 add untagged 9-13

Создать IP-интерфейсы:

create ipif 1000 192.168.1.2/24 v1000 state enable

```
DGS-3120-24TC:admin#show dhcp_relay
Command: show dhcp_relay

DHCP/BOOTP Relay Status      : Enabled
DHCP/BOOTP Hops Count Limit   : 16
DHCP/BOOTP Relay Time Threshold : 0
DHCP Vendor Class Identifier Option 60 State: Disabled
DHCP Client Identifier Option 61 State: Disabled
DHCP Relay Agent Information Option 82 State : Disabled
DHCP Relay Agent Information Option 82 Check : Disabled
DHCP Relay Agent Information Option 82 Policy : Replace
DHCP Relay Agent Information Option 82 Remote ID : default
DHCP Relay Agent Information Option 82 Circuit ID : default

Interface     Server 1        Server 2        Server 3        Server 4
-----
1000          192.168.1.1

Server          VLAN ID List
-----
192.168.1.1    2000,3000
```

Рис. 9.59. Настройки DHCP relay-агента

```
create ipif 2000 192.168.2.2/24 v2000 state enable
create ipif 3000 192.168.3.2/24 v3000 state enable
```

Включить DHCP Relay:

```
enable dhcp_relay
```

Указать IP-адрес DHCP-сервера:

```
config dhcp_relay add ipif 1000 192.168.1.1
```

Настроить передачу DHCP-пакетов из VLAN v2000 и v3000:

```
config dhcp_relay add vlanid 2000, 3000 192.168.1.1
```

9.4.5. Опция DHCP Relay Agent Information (Option 82)

Опция *DHCP Relay Agent Information (Option 82)* преимущественно используется в сетях провайдеров и больших корпоративных сетях. Relay-агент добавляет дополнительную информацию (опцию 82) в сообщения DHCP, (рис. 9.60) передаваемые клиентом DHCP-серверу. Эта информация позволяет идентифицировать точку подключения клиента.

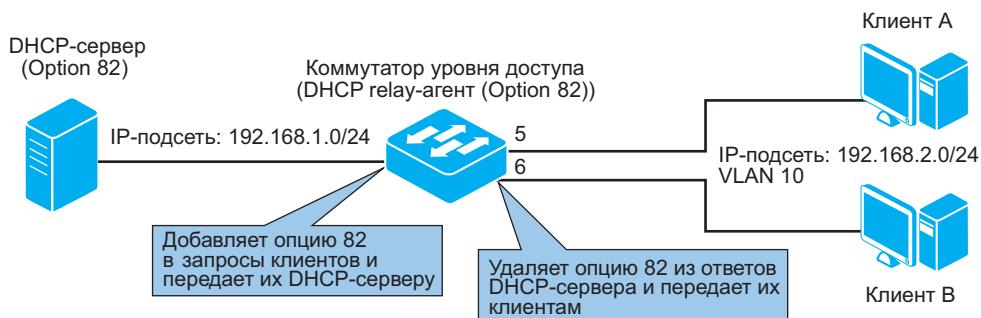


Рис. 9.60. Использование DHCP relay-агента с поддержкой опции 82 в сети

Как известно, relay-агент вставляет свой IP-адрес в поле *Relay Agent IP address* сообщения DHCP, полученного от клиента. Используя этот IP-адрес, сервер выбирает нужный пул адресов. Опция 82 уточняет запрос, позволяя серверу выбирать поддиапазон или какой-то конкретный адрес из пула на основе информации о маршрутизаторе/коммутаторе и номере его порта, который клиент использует для подключения к сети.

Формат опции *Relay Agent Information* (рис. 9.61) определен в RFC 3046. Она начинается с поля *Code*, имеющего значение 82, за которым следует поле *Length*, определяющее общую длину опции. За полем *Length* расположено

Code	Length	Agent Information Field					
82	N	i1	i2	i3	iN	

Рис. 9.61. Формат опции Relay Agent Information

9. Протоколы уровня приложений

поле *Agent Information Field*, которое состоит из последовательности подопций, состоящих из трех полей: *SubOpt* (код подопции)/*Length* (длина подопции)/*Value* (значение подопции).

В RFC 3046 определены только две подопции:

- *Agent Circuit ID* (SubOpt 1) — строка, идентифицирующая интерфейс relay-агента, принимающий DHCP-сообщения от клиента;
- *Agent Remote ID* (SubOpt 2) — строка, идентифицирующая relay-агента.

Однако некоторые производители предпочитают использовать их собственные расширения опции 82. RFC 4342 описывает подопции опции 82, в которых передается информация, специфичная для производителя.

Подопции *Agent Circuit ID* и *Agent Remote ID* обычно определяются клиентским устройством доступа и зависят от конфигурации сети. Программное обеспечение коммутаторов D-Link с поддержкой функции DHCP relay-агента позволяет конфигурировать формат этих подопций. По умолчанию в качестве *Agent Remote ID* используется MAC-адрес коммутатора. В качестве *Agent Circuit ID* по умолчанию используется строка, имеющая формат, показанный на рис. 9.62. Другие поддерживающие форматы *Agent Remote ID* и *Agent Circuit ID* описаны в руководствах пользователей соответствующих моделей коммутаторов.

a.	b.	c.	d.	e.	f.	g.
1	0x6	0	4	VLAN	Module ID	Port ID
1 байт	1 байт	1 байт	1 байт	2 байта	1 байт	1 байт

- Sub-option type* (тип подопции): значение 1 для Circuit ID.
- Length* (длина): значение длины должно быть 6.
- Circuit ID's sub-option* (подопция Circuit ID): значение должно быть 0.
- Sub-option's length* (длина подопции): значение должно быть 4.
- VLAN ID (S-VID)*: идентификатор VLAN.
- Module ID* (идентификатор модуля): для автономного коммутатора значение равно 0; для стекируемого указывается Box ID.
- Port ID* (идентификатор порта): номер порта.

Рис. 9.62. Формат *Agent Circuit ID* по умолчанию

Чтобы использовать опцию 82 в сети, ее поддержка должна быть реализована и на relay-агенте, и на DHCP-сервере. Если DHCP-сервер не поддерживает опцию 82, то соответствующие поля в DHCP-сообщениях будут им игнорироваться. Если сервер распознает опцию 82, он может использовать информацию из нее для реализации политики назначения IP-адресов и других параметров (рис. 9.63). В примере, показанном на рис. 9.64, сервер с поддержкой опции 82 присваивает клиенту, подключенному к порту 5 коммутатора с MAC-адресом AC:F1:DF:B5:FC:00, IP-адрес 192.168.2.5, а клиенту, подключенному к порту 6, — IP-адрес 192.168.2.6.

Технологии TCP/IP в современных компьютерных сетях

```
Ethernet II, Src: D-LinkIn_b5:fc:00 (ac:f1:df:b5:fc:00), Dst: D-LinkIn_1f:9c:01 (84:c9:b2:1f:9c:01)
Internet Protocol Version 4, Src: 192.168.2.2, Dst: 192.168.1.1
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Discover)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 1
  Transaction ID: 0xdcda1641
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 192.168.2.2
  Client MAC address: Giga-Byt_5e:d3:87 (fc:aa:14:5e:d3:87)
  Client hardware address padding: 000000000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Discover)
  Option: (61) Client identifier
  Option: (12) Host Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (82) Agent Information Option
    Length: 18
    Option 82 Suboption: (1) Agent Circuit ID
      Length: 6
      Agent Circuit ID: 000407d00005
    Option 82 Suboption: (2) Agent Remote ID
      Length: 8
      Agent Remote ID: 0006acf1dfb5fc00
  Option: (255) End
```

Рис. 9.63. Сообщение DHCPDISCOVER с опцией 82, перенаправленное relay-агентом DHCP-серверу

DHCP-сервер сохраняет опцию 82 в ответе клиенту. Когда relay-агент получает сообщение от сервера, он удаляет опцию 82 из ответа и пересыпает его клиенту. При этом relay-агент пересыпает ответ сервера только на тот порт, к которому подключен DHCP-клиент (т. е. одноадресно). Это позволяет исключить рассылку широковещательного трафика по всему широковещательному домену.

На пути между первичным relay-агентом (к которому непосредственно подключен клиент) и DHCP-сервером могут лежать промежуточные relay-агенты (рис. 9.65). Если промежуточные relay-агенты не поддерживают опцию 82, они должны пересыпать DHCP-сообщения, несмотря на то, содержится ли в них опция 82 или нет.

Relay-агенты с поддержкой опции 82 могут выполнять три политики перенаправления DHCP-сообщений серверу.

1. **Drop:** политика отбрасывания DHCP-сообщений, которые уже содержат опцию 82. Этую политику удобно применять на пограничных relay-агентах для защиты от подключений неавторизованных клиентов, которые рассыпают DHCP-сообщения с недействительной опцией 82. Если DHCP-сообщение

```
$ sudo gedit /etc/dhcp/dhcpd.conf

class "port-5" {
    match if binary-to-ascii (10, 8, "", suffix(option agent.circuit-id, 1)) = "5" and
binary-to-ascii (16, 8, ":" , substring(option agent.remote-id, 2, 6)) = "ac:f1:df:b5:fc:0";
}

class "port-6" {
    match if binary-to-ascii (10, 8, "", suffix(option agent.circuit-id, 1)) = "6" and
binary-to-ascii (16, 8, ":" , substring(option agent.remote-id, 2, 6)) = "ac:f1:df:b5:fc:0";
}

subnet 192.168.2.0 netmask 255.255.255.0 {
    pool {
        allow members of "port-5";
        range 192.168.2.5;
        option subnet-mask 255.255.255.0;
        option routers 192.168.2.2;
        default-lease-time 600;
        max-lease-time 7200;
    }
    pool {
        allow members of "port-6";
        range 192.168.2.6;
        option subnet-mask 255.255.255.0;
        option routers 192.168.2.2;
        default-lease-time 600;
        max-lease-time 7200;
    }
}
```

Рис. 9.64. Пример настройки политики назначения параметров TCP/IP на DHCP-сервере с поддержкой опции 82

не содержит опцию 82, она будет в него добавлена при пересылке DHCP-серверу.

2. **Keep:** политика сохранения опции 82 в полученном DHCP-сообщении. Если relay-агент получает DHCP-сообщение, содержащее опцию 82, оно будет без изменений перенаправлено DHCP-серверу. Если DHCP-сообщение не содержит опцию 82, она будет в него добавлена при пересылке DHCP-серверу.

3. **Replace:** политика замены информации в опции 82 полученного DHCP-сообщения. Если relay-агент получает DHCP-сообщение, содержащее опцию 82, она будет заменена новой.

Информация опции 82, содержащаяся в ответе DHCP-сервера клиенту должна быть идентичной информации, полученной в запросе клиента. На relay-агенте можно настроить функцию проверки опции 82 в ответах DHCP-сервера. По умолчанию она отключена. Когда функция активирована, relay-агент будет проверять наличие опции 82 и ее содержимое в DHCP-сообщениях, получаемых от сервера. Если опция 82 отсутствует в ответе или ее содержимое отличается от информации, первоначально добавленной relay-агентом (проверяется подопция *Agent Remote ID*), то DHCP-сообщение отбрасывается. В противном случае relay-агент удаляет опцию и пересыпает DHCP-сообщение клиенту.

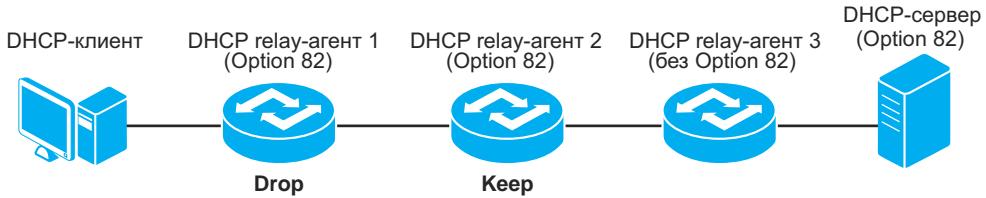


Рис. 9.65. Множество relay-агентов на пути передачи сообщения от DHCP-клиента DHCP-серверу

Активизация проверки позволяет повысить защиту сети от получения DHCP-ответов от ненадежных источников или с недостоверной информацией в опции 82.

Если проверка не производится, relay-агент будет удалять опцию 82 и пересыпал DHCP-сообщение клиенту.

9.4.6. Функция DHCP Local Relay

Из определения DHCP relay-агента следует, что это узел, который настроен для передачи пакетов DHCP между клиентом и сервером, находящихся в разных сетях. Relay-агент помимо своего IP-адреса, может добавлять в запрос клиента информацию опции 82. Предположим, что в пределах локальной сети или подсети администратору надо дифференцировать выдачу IP-адресов клиентам. Например, клиентам, подключенным к портам 5 и 6 коммутатора, всегда должны выдаваться строго определенные адреса, остальным клиентам — свободные адреса из какого-то пула. Данную задачу можно решить с помощью функции DHCP Local Relay (рис. 9.66), поддерживаемой коммутаторами D-Link. Она позволяет коммутатору добавлять опцию 82 в запросы клиентов, которые находятся с DHCP-сервером в одной IP-сети/подсети (в одном широковещательном домене). При этом локальный

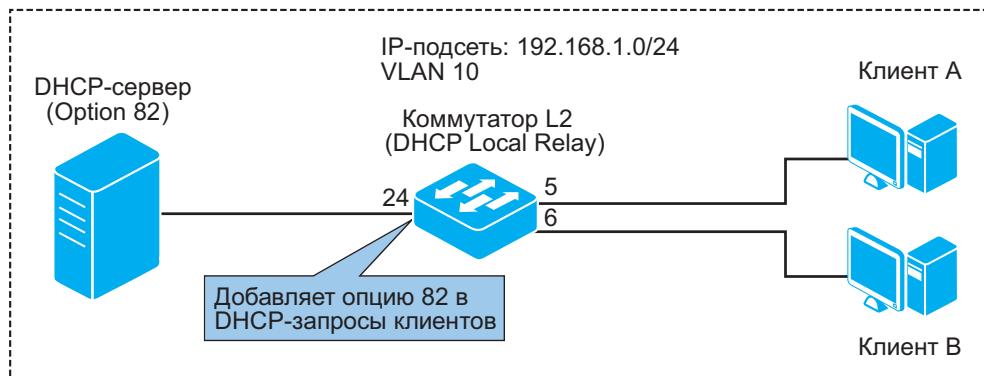


Рис. 9.66. Использование функции DHCP Local Relay в сети

9. Протоколы уровня приложений

relay-агент не изменяет IP- и MAC-адреса приемника и назначения в DHCP-сообщении, а также не добавляет свой адрес в поле *Relay Agent IP address*. В DHCP-сообщение клиента будет автоматически добавляться опция 82 (рис. 9.67). На основании информации из опции сервер будет определять

```
Ethernet II, Src: WistronI_72:a5:82 (20:6a:8a:72:a5:82), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Bootstrap Protocol (Discover)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xa59b753e
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: WistronI_72:a5:82 (20:6a:8a:72:a5:82)
  Client hardware address padding: 000000000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Discover)
  Option: (61) Client identifier
  Option: (12) Host Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (82) Agent Information Option
    Length: 18
    Option 82 Suboption: (1) Agent Circuit ID
      Length: 6
      Agent Circuit ID: 000403e80002
    Option 82 Suboption: (2) Agent Remote ID
      Length: 8
      Agent Remote ID: 0006acf1dfb5fc00
  Option: (255) End
```

Рис. 9.67. Сообщение DHCPDISCOVER с опцией 82, добавленной локальным relay-агентом

требуемые параметры, руководствуясь настроенной на нем политикой выдачи адресов. Опция 82 не будет удаляться локальным relay-агентом из ответов DHCP-сервера.

9.4.7. Технология DHCP Snooping

Протокол DHCP является одним из наиболее используемых протоколов. Однако в нем не реализованы методы защиты и проверки подлинности пакетов. В связи с этим он может быть подвергнут разного рода атакам. Например, таким, как исчерпание диапазона IP-адресов (атака DHCP starvation) и установка несанкционированного DHCP-сервера. При атаке

DHCP starvation злонамеренный клиент попытается загрузить сервер и захватить весь диапазон IP-адресов, предназначенный для подлинных клиентов сети. Для этого ему потребуется просто сгенерироваться уникально идентифицируемые пакеты. Он может использовать произвольные MAC-адреса источника и посыпать сообщения DHCPDISCOVER с этими ложными MAC-адресами серверу. Чем больше сообщений DHCPDISCOVER будет отправлено, тем больше времени потребуется серверу на их обработку и отправку предложений DHCPOFFER. Поскольку DHCP-сервер не может отличить злонамеренного клиента от подлинного, он будет выделять адреса из диапазона ложным клиентам. Злонамеренный клиент не будет отправлять серверу запросы DHCPREQUEST, поэтому сервер будет их ждать. IP-адрес будет считаться зарезервированным, но не выделенным, потому что процедура получения параметров не завершена. В результате этой атаки подлинные клиенты не смогут динамически получить IP-адреса и другие параметры.

Обнаружить эту атаку несложно. Ее индикатором является внезапное увеличение количества динамически изученных MAC-адресов на одном из портов коммутатора. При нормальных условиях работы сети порт коммутатора обычно динамически изучает от одного до трех MAC-адресов.

Механизмом, который позволяет коммутаторам контролировать работу протокола DHCP, является технология *DHCP Snooping*. Она выполняет углубленную проверку DHCP-сообщений, приходящих на порты коммутатора, и динамически создает таблицу привязки DHCP Snooping (DHCP Snooping binding database). Запись таблицы (binding entry) представляет собой привязку IP- и MAC-адресов клиента к порту, через который он подключен. В дальнейшем эта таблица может использоваться совместно с механизмами Dynamic ARP Inspection и IP Source Guard/IP Inspection.

DHCP Snooping вводит понятия *недоверенных* (untrusted) и *доверенных* (trusted) портов. По умолчанию все порты коммутатора являются недоверенными, что позволяет защитить клиентов от получения IP-адресов от несанкционированного DHCP-сервера. Работа DHCP-сервера при подключении к такому порту будет невозможна. В качестве доверенного порта обычно конфигурируется тот порт, к которому подключен DHCP-сервер или другой коммутатор. Недоверенными являются все остальные порты. Таким образом, DHCP Snooping можно рассматривать как специализированный межсетевой экран, расположенный между доверенными и недоверенными портами.

Таблица привязки DHCP Snooping создается только для *недоверенных портов*. Выполняя анализ DHCP-пакетов, коммутатор узнает IP-адрес, назначенный DHCP-сервером клиенту, подключенному к определенному порту. Он динамически создает запись, связывающую следующие параметры: IP-адрес, MAC-адрес, время аренды, порт. IP-адрес — это адрес, назначенный DHCP-сервером клиенту; MAC-адрес — это MAC-адрес клиента; порт — идентифицирует порт, к которому подключен клиент; время аренды — период времени, на который DHCP-сервером выделен IP-адрес.

9. Протоколы уровня приложений

Новая запись (если она не существует) создается коммутатором после получения сообщения DHCPACK. Сообщение DHCPACK отправляется DHCP-сервером, когда он подтвердил назначение клиенту IP-адреса. Коммутатор удаляет запись из таблицы (если она существовала) после получения сообщения DHCPNAK, DHCPDECLINE или DHCPRELEASE. Напомним, что DHCP-сервер отправляет клиенту отрицательное подтверждение DHCPNAK, если его запрос некорректный или адрес уже используется. Если полученный адрес не уникален, клиент шлет серверу сообщение DHCPDECLINE. Когда клиент решает отказаться от IP-адреса, он посыпает серверу сообщение DHCPRELEASE.

Для повышения безопасности администратор может ограничить максимальное количество создаваемых в процессе автоизучения записей IP-MAC на порт. Также можно ограничить количество сообщений DHCP, которое порт будет получать в секунду. При превышении лимита состояние порта изменится с active на error disable. Таким образом можно будет предотвратить DoS-атаку, вызванную отправкой непрерывного потока DHCP-сообщений.

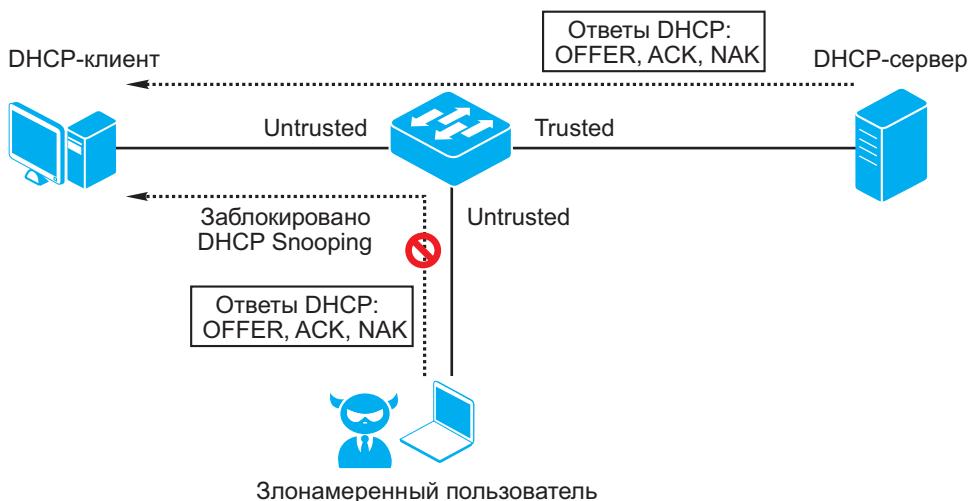


Рис. 9.68. Доверенные и недоверенные порты DHCP Snooping

Стоит отметить, что для тех узлов, которые не используют DHCP, записи таблицы можно создавать вручную. Динамически записи для устройств со статическими IP-адресами создаваться не будут.

Реализация технологии DHCP Snooping на коммутаторах D-Link возможна в двух вариантах. В первом варианте она является режимом функции *IP-MAC-Port Binding* (IMPB); во втором — самостоятельной функцией. Рассмотрим различия реализаций.

Switch(config)#sh ip dhcp snooping binding					
MAC Address	IP Address	Lease(seconds)	Type	VLAN	Interface
20-6A-8A-72-A5-82	192.168.1.3	416	dhcp-snooping	1	eth1/0/6
FC-AA-14-5E-D3-87	192.168.1.4	588	dhcp-snooping	1	eth1/0/16
Total Entries: 2					

Рис. 9.69. Пример таблицы привязки DHCP Snooping

Внимание: реализация технологии DHCP Snooping может незначительно отличаться в разных сериях коммутаторов D-Link. За информацией о настройке конкретной модели коммутаторов обращайтесь к технической документации.

Функция IP-MAC-Port Binding (IMPB), реализованная в коммутаторах D-Link, позволяет контролировать доступ компьютеров в сеть на основе их IP/MAC-адресов и порта подключения. Она также позволяет бороться с атаками типа ARP Spoofing, во время которых злонамеренные пользователи перехватывают трафик или прерывают соединение, манипулируя пакетами ARP. Функция включает четыре режима работы: ARP Inspection (по умолчанию), IP Inspection, ND Snooping и DHCP/DHCPv6 Snooping (рис. 9.68).

Работа функции основана на сравнении параметров входящих пакетов с параметрами хранящихся на коммутаторе записей, связывающих MAC- и IP-адреса клиентских устройств с портами подключения. В случае совпадения всех составляющих (IP/MAC-адресов и порта) пакеты будут передаваться, и клиенты получат доступ в сеть. Если при подключении клиента связка MAC-IP-порт будет отличаться от параметров заранее сконфигурированной записи, коммутатор заблокирует MAC-адрес соответствующего узла с занесением его в «черный лист».

Режим DHCP Snooping используется коммутатором для динамического создания записей IP-MAC на основе анализа DHCP-пакетов (рис. 9.69) и привязки их к портам с включенной функцией IMPB (администратору не требуется создавать записи вручную). Таким образом, коммутатор автоматически создает «белый лист» IMPB в таблице коммутации и/или таблице ACL (если включен режим IP inspection). При этом для обеспечения корректной работы сервер DHCP или другой коммутатор должен быть подключен к доверенному порту с выключенной функцией IMPB. В случае подключения DHCP-сервера или коммутатора к порту, на котором включена функция IMPB, для него необходимо создать статическую связку IP-MAC-порт. В противном случае пакеты будут отбрасываться.

Внимание: чтобы проводилась проверка входящих пакетов, режим DHCP Snooping должен использоваться совместно с режимами ARP Inspection или IP Inspection.

9. Протоколы уровня приложений

Рассмотрим пример настройки функции IMPB в режиме DHCP Snooping на коммутаторе DGS-3120-24TC; схема сети показана на рис. 9.70.

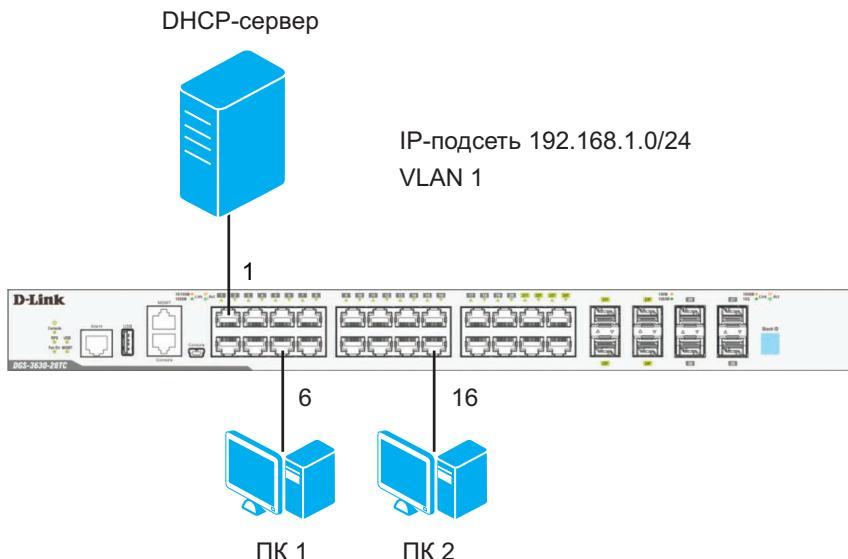


Рис. 9.70. Схема сети

- Активировать функцию IMPB на портах:

```
config address_binging ip_mac ports 2-24 arp_inspection strict
```

- Указать максимальное количество записей в процессе автоизучения записей IP-MAC на порт:

```
config address_binding dhcp_snoop max_entry ports 2-24 limit 1
```

- Активировать режим DHCP Snooping:

```
enable address_binding dhcp_snoop
```

В сериях коммутаторов DGS-3130 и DGS-3630 технология DHCP Snooping реализована как самостоятельная функция. Ее основное отличие от режима DHCP Snooping заключается в том, что дополнительно к построению таблицы привязки недоверенный порт может выполнять проверку принимаемых DHCP-сообщений без активизации дополнительных функций или режимов. Другими словами, функция DHCP Snooping выполняет и проверку DHCP-сообщений, и динамическое создание записей.

Недоверенный порт будет отбрасывать DHCP-сообщения в следующих случаях.

1. Если получены сообщения DHCPOFFER, DHCPACK и DHCPNAK. Понятно, что DHCP-клиенты не отправляют сообщения DHCPOFFER или DHCPACK. Они рассыпают только сообщения DHCPDISCOVER и DHCPREQUEST.

2. Если в поле *Relay Agent IP address* DHCP-сообщения указан адрес, отличный от 0.0.0.0, или имеется опция 82.

3. При несовпадении MAC-адреса источника в заголовке Ethernet с MAC-адресом клиента, указанным в поле *Client MAC address* сообщения DHCP, если администратором была настроена проверка MAC-адреса.

4. Если от узла, для которого имеется запись в таблице привязки DHCP Snooping, получены сообщения DHCPRELEASE, DHCPCDECLINE, но номер порта, который их принял, отличается от номера порта, указанного в таблице.

На доверенных портах проверка входящих DHCP-сообщений проводиться не будет.

После того как для соответствующего недоверенного порта создана запись, коммутатор начинает сравнивать параметры входящих DHCP-сообщений с записями в таблице DHCP Snooping. DHCP-пакеты, в которых не совпадает хотя бы один параметр, будут коммутатором отбрасываться.

Рассмотрим пример настройки функции DHCP Snooping на коммутаторе DGS-3630-28TC в сети, показанной на рис. 9.70.

- Активировать функцию DHCP Snooping глобально на коммутаторе:

```
ip dhcp snooping
```

- Включить функцию DHCP Snooping в VLAN:

```
ip dhcp snooping vlan 1
```

- Настроить доверенный порт, к которому подключен DHCP-сервер:

```
interface ethernet 1/0/1
```

```
ip dhcp snooping trust
```

- Указать максимальное количество записей в таблице DHCP Snooping на порт:

```
interface range ethernet 1/0/2-24
```

```
ip dhcp snooping limit entries 1
```

Функция DHCP Snooping может использоваться совместно с функциями Dynamic ARP Inspection (DAI) и IP Source Guard для защиты от атак IP/ARP Spoofing.

9.5. Протокол DHCPv6

Напомним, что в протоколе IPv6 определены два механизма автоконфигурации адресов: Stateless autoconfiguration (RFC 4862) и Stateful autoconfiguration (RFC 3315).

Stateless autoconfiguration позволяет узлам генерировать свой собственный адрес на основе комбинации локально доступной информации и информации, объявляемой маршрутизаторами. Маршрутизаторы объявляют префиксы, определяющие подсеть (или подсети), а узлы самостоятельно генерируют идентификаторы интерфейсов. В отсутствии маршрутизатора узлы могут автоматически генерировать адрес Link-Local IPv6 Unicast.

9. Протоколы уровня приложений

Stateful autoconfiguration позволяет узлам получать адрес интерфейса и/или конфигурационные параметры с помощью протокола DHCPv6.

Механизмы автоконфигурации stateless и stateful могут дополнять друг друга и использоваться совместно.

Протокол *Dynamic Host Configuration Protocol for IPv6* (DHCPv6) позволяет DHCP-серверам передавать конфигурационные параметры узлам с поддержкой IPv6. Операции DHCPv6 отличаются от операций DHCPv4, так как протокол был полностью переписан. Он не основан на DHCPv4 или BOOTP, за исключением концептуальных терминов. Оба протокола используют концепцию DHCP-сервера, DHCP-клиента и relay-агента. Если клиент хочет получить конфигурационные параметры, он отправляет запрос в локальную сеть. Сервер может получить этот запрос напрямую или через relay-агента. В остальном протоколы несовместимы.

DHCPv6 может использоваться или для получения IPv6-адреса и конфигурационных параметров, или только для получения дополнительных конфигурационных параметров, когда клиент уже имеет IPv6-адрес. В первом случае DHCPv6 называется «Stateful address autoconfiguration protocol» или «Stateful autoconfiguration protocol». Он описан в RFC 3315. Во втором случае DHCPv6 называется «Stateless Dynamic Host Configuration Protocol» и описывается в RFC 3736. Другими словами, **Stateful DHCPv6** позволяет DHCP-серверам передавать клиентам конфигурационные параметры, такие как префиксы IPv6, адреса IPv6. Сервер должен отслеживать состояние клиентов и выполнять его обновление. **Stateless DHCPv6** служит для получения дополнительных конфигурационных параметров, таких как адреса DNS-серверов, и не требует отслеживания состояния клиентов. Узел, который использует Stateless DHCPv6, должен получить IPv6-адрес через другой механизм, обычно через stateless autoconfiguration.

DHCPv6 использует для рассылки сообщений уникальные и групповые адреса. Для использования в DHCPv6 зарезервированы следующие групповые адреса:

- **All_DHCP_Relay_Agents_and_Servers** (FF02::1:2): идентифицирует группу, включающую в себя все серверы и relay-агенты DHCPv6 в пределах области Link-Local. Адрес используется клиентом для взаимодействия с соседними (подключенными к тому же каналу) серверами и relay-агентами;

- **All_DHCP_Servers** (FF05::1:3): идентифицирует группу всех DHCPv6-серверов в пределах области Site-Local. Адрес используется relay-агентом для взаимодействия с серверами, если он хочет отправить сообщение всем серверам или если он не знает индивидуального адреса сервера.

В качестве протокола транспортного уровня DHCPv6 использует UDP. При этом применяются два номера порта: порт 547 — для отправки сообщений от клиента серверу и порт 546 — для отправки сообщений от сервера клиенту.

9.5.1. Типы сообщений DHCPv6

Типы сообщений, определенные в DHCPv6, перечислены в табл. 9.4.

Таблица 9.4. Типы сообщений DHCPv6

Сообщение	Описание
SOLICIT (1)	Клиент отправляет Solicit, чтобы обнаружить серверы
ADVERTISE (2)	Сервер отправляет сообщение Advertise, чтобы объявить о доступном сервисе DHCP в ответ на сообщение Solicit, полученное от клиента
REQUEST (3)	Клиент отправляет сообщение Request, чтобы запросить у определенного сервера конфигурационные параметры, включая IP-адреса
CONFIRM (4)	Клиент отправляет сообщение Confirm любому доступному серверу, чтобы определить, является ли адрес, который ему был назначен, действительным для канала, к которому он подключен. Другими словами, сообщение Confirm используется клиентом для определения того, к какому каналу он подключен: к прежнему или новому
RENEW (5)	Клиент отправляет сообщение Renew серверу, который первоначально предоставил ему адрес и конфигурационные параметры для продления аренды адреса и обновления конфигурационных параметров
REBIND (6)	Клиент отправляет сообщение Rebind любому доступному серверу, чтобы продлить время аренды адреса и других конфигурационных параметров; это сообщение отправляется, если не пришел ответ от сервера на сообщение Renew
REPLY (7)	Сервер отправляет сообщение Reply: содержащее назначенные адреса и конфигурационные параметры в ответ на сообщение Solicit, Request, Renew или Rebind, полученное от клиента; содержащее конфигурационные параметры в ответ на сообщение Information-request; содержащее подтверждение или отказ в том, что адрес клиента действителен на канале, к которому он подключен в ответ на сообщение Confirm; для подтверждения получения сообщения Release или Decline
RELEASE (8)	Клиент отправляет сообщение Release серверу, который назначил ему адреса, чтобы сообщить, что один или несколько назначенных адресов им больше не используются

9. Протоколы уровня приложений

Окончание табл. 9.4

Сообщение	Описание
DECLINE (9)	Клиент отправляет сообщение Decline серверу, чтобы сообщить, что один или несколько назначенных сервером адресов уже используются на канале, к которому он подключен
RECONFIGURE (10)	Сервер отправляет сообщение Reconfigure, чтобы уведомить клиента о появлении новых или обновлении прежних конфигурационных параметров. Клиент должен инициировать обмен сообщениями Renew/Reply или Information-request/Reply с сервером для обновления информации
INFORMATION-REQUEST (11)	Клиент отправляет сообщение Information-request серверу с целью запроса конфигурационных параметров без назначения каких-либо IPv6-адресов
RELAY-FORW (12)	Relay-агент отправляет сообщение Relay-forward для перенаправления сообщений серверу напрямую или через другого relay-агента. Сообщение, полученное от клиента или другого relay-агента, помещается в опцию сообщения Relay-forward
RELAY-REPL (13)	Сервер отправляет relay-агенту сообщение Relay-reply, содержащее сообщение, которое relay-агент должен доставить клиенту. Сообщение Relay-reply может быть перенаправлено несколькими relay-агентами для доставки целевому relay-агенту. Сервер инкапсулирует сообщение для клиента в опцию сообщения Relay-reply. Далее relay-агент извлечет его и передаст клиенту

Все сообщения DHCPv6, отправляемые между клиентом и сервером, имеют одинаковый фиксированный формат заголовка и переменный формат поля опций (рис. 9.71).

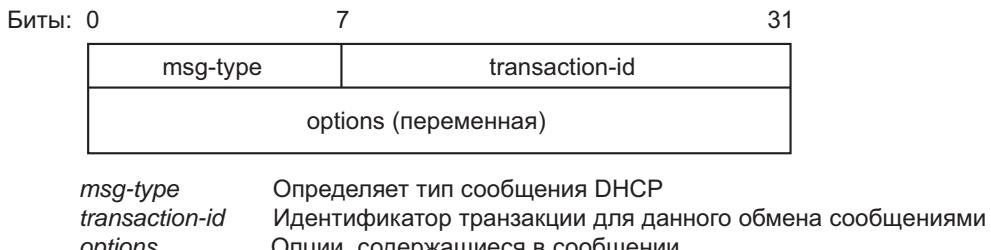


Рис. 9.71. Формат сообщения DHCPv6

В DHCPv6 опции используются для обмена дополнительной информацией между клиентом и сервером. Формат опции DHCPv6 включает три поля: *option-code* — идентифицирует опцию; *option-len* — указывает длину поля *option-data* в байтах; *option-data* — содержит данные опции.

В качестве примера приведем некоторые опции DHCPv6:

- *Client Identifier* (1): используется для передачи DUID, идентифицирующего клиента;
- *Server Identifier* (2): используется для передачи DUID, идентифицирующего сервер;
- *Identity Association for Non-temporary Addresses* (3): используется для передачи параметров, связанных с ассоциацией идентичности для постоянных адресов;
- *Option Request* (6): используется для определения списка опций, запрашиваемых клиентом;
- *Preference* (7): отправляется сервером клиенту, чтобы повлиять на клиента при выборе сервера;
- *Relay Message* (9): содержит перенаправляемое сообщение DHCPv6;
- *Status Code* (13): возвращает состояние, относящееся к DHCP-сообщению или опции, в которой оно появилось;
- *Rapid Commit* (14): используется для извещения о том, что для назначения адреса используется обмен двумя сообщениями;
- *Interface-id* (18): используется relay-агентом DHCPv6 для идентификации интерфейса, на который было получено сообщение от клиента. Аналогична подопции *Agent Circuit ID* опции 82 протокола DHCPv4;
- *Relay Agent Remote-ID* (37): содержит строку, идентифициирующую relay-агента DHCPv6. Аналогична подопции *Agent Remote ID* опции 82 протокола DHCPv4;
- *DNS Recursive Name Server* (23): предоставляет список из одного или нескольких IPv6-адресов рекурсивных серверов DNS, к которым DNS-преобразователь клиента может отправлять запросы;
- *Domain Search List* (24): определяет список поиска доменов, который клиент использует в процессе разрешения имен с помощью DNS;

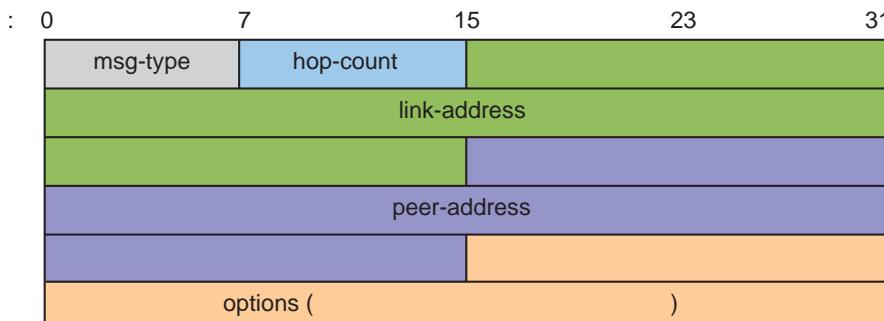


Рис. 9.72. Общий формат сообщений RELAY

- *Identity Association for Prefix Delegation* (25): используется для передачи ассоциации идентичности для делегирования префикса (prefix delegation identity association, IA_PD), параметров и префиксов, связанных с ней;
- *IA_PD Prefix* (26): используется для указания префиксов IPv6-адресов, ассоциированных с IA_PD.

Relay-агент может отправлять сообщения двух типов (RELAY-FORW и RELAY-REPL), которые используют общий формат, показанный на рис. 9.72.

В сообщении Relay-forward поля имеют следующие значения:

<i>msg-type</i>	тип сообщения RELAY-FORW;
<i>hop-count</i>	количество relay-агентов, которые перенаправили это сообщение;
<i>link-address</i>	global или site-local адрес, который будет использоваться сервером для идентификации канала, к которому подключен клиент;
<i>peer-address</i>	адрес клиента или relay-агента, от которого получено перенаправляемое сообщение;
<i>options</i>	поле должно включать опцию Relay Message.

В сообщении Relay-reply поля имеют следующие значения:

<i>msg-type</i>	тип сообщения RELAY-REPL;
<i>hop-count</i>	копируется из сообщения Relay-forward;
<i>link-address</i>	копируется из сообщения Relay-forward;
<i>peer-address</i>	копируется из сообщения Relay-forward;
<i>options</i>	поле должно включать опцию Relay Message.

9.5.2. Уникальный идентификатор DHCP (DUID)

Каждый клиент и сервер DHCPv6 имеет уникальный идентификатор DUID (DHCP Unique Identifier). Сервер использует DUID для идентификации клиента при выборе конфигурационных параметров для IA (Identity Association). Клиент использует DUID для идентификации сервера в передаваемых сообщениях.

DUID передается как опция, поскольку он может иметь переменную длину и его присутствие требуется не во всех сообщениях. Для каждого клиента и сервера идентификатор должен быть уникальным и постоянным, т. е. не изменяемым в течение времени (например, при аппаратных изменениях DUID устройства не должен изменяться).

DUID состоит из двухоктетного типа кода и следующего за ним переменного числа байтов, которые непосредственно содержат идентификатор. Длина идентификатора не может превышать 128 байт.

В настоящее время определены следующие типы кодов, указывающие формат идентификатора:

1. Link-layer address plus time (DUID-LLT);
2. Vendor-assigned unique ID based on Enterprise Number (DUID-EN);
3. Link-layer address (DUID-LL).

Формат Link-layer address plus time (рис. 9.73) состоит из двухбайтного поля *Type*, содержащего значение 1; двухбайтного поля *Hardware type*, содержащего код типа оборудования, назначенного IANA; четырехбайтного поля *Time*, содержащего значение времени; адреса канального уровня (*Link-layer address*) одного любого сетевого интерфейса, который подключен к устройству DHCP во время генерирования DUID. Значение времени — это время, когда генерировался DUID, представленное в секундах с полуночи (UTC) 1 января 2000 года, по модулю 2^{32} . Другими словами, идентификатор генерируется из адреса канального уровня интерфейса и текущего времени. Для Ethernet адресом канального уровня является MAC-адрес.

```
Client Identifier
  Option: Client Identifier (1)
  Length: 14
  Value: 0001000116d50715206a8a737c8c
  DUID: 0001000116d50715206a8a737c8c
  DUID Type: link-layer address plus time (1)
  Hardware type: Ethernet (1)
  DUID Time: Feb 20, 2012 16:28:53.000000000 RTZ 2 (зима)
  Link-layer address: 20:6a:8a:73:7c:8c
```

Рис. 9.73. Формат Link-layer address plus time

Формат Vendor-assigned unique ID based on Enterprise Number состоит из двухбайтного поля *Type*, содержащего значение 2; поля *Private Enterprise Number*, содержащего личный номер организации, зарегистрированный IANA; поля *Unique Identifier*, содержащего уникальный идентификатор, назначенный устройству производителем. Другими словами, идентификатор назначается производителем.

Формат Link-layer address (рис. 9.74) состоит из двухбайтного поля *Type*, содержащего значение 3; двухбайтного поля *Hardware type*, содержащего код типа оборудования, назначенного IANA; адреса канального уровня (*Link-layer address*) одного любого сетевого интерфейса, который постоянно подключен к клиенту или серверу. Другими словами, идентификатор генерируется из адреса канального уровня интерфейса.

```
Server Identifier
  Option: Server Identifier (2)
  Length: 10
  Value: 0003000684c9b21f9c00
  DUID: 0003000684c9b21f9c00
  DUID Type: link-layer address (3)
  Hardware type: IEEE 802 (6)
  Link-layer address: 84:c9:b2:1f:9c:00
```

Рис. 9.74. Формат Link-layer address

9.5.3. Ассоциация идентичности (IA)

Ассоциация идентичности (Identity Association, IA) — это структура, через которую сервер и клиент могут идентифицировать, группировать и управлять набором связанных IPv6-адресов. Каждая IA состоит из идентификатора ассоциации идентичности (Identity Association Identifier, IAID) (рис. 9.75) и ассоциированной конфигурационной информации. Каждый клиент должен ассоциировать IA как минимум с одним своим интерфейсом, для которого он запрашивает назначение IPv6-адресов у DHCP-сервера. Идентификатор IAID выбирает клиент, и он должен уникально идентифицировать каждую IA.

Другими словами, клиент использует ассоциацию идентичности для идентификации каждого интерфеса, который конфигурируется через DHCPv6. IA интерфейса состоит из его конфигурационных параметров и IAID. Когда клиент запрашивает у сервера параметры для определенного интерфейса, он включает в запрос IAID для его идентификации.

Адаптер беспроводной локальной сети Беспроводное сетевое соединение:	
DNS-суффикс подключения	:
Описание	: Broadcom 4313 802.11b/g/n
Физический адрес	: CC-52-AF-0A-9C-E1
DHCP включен	: Да
Автонастройка включена	: Да
Локальный IPv6-адрес канала	: fe80::f1a8:117d:6b9:74a4%13(Основной)
IPv4-адрес	: 192.168.100.3(Основной)
Маска подсети	: 255.255.255.0
Аренда получена	: 19 апреля 2018 г. 9:08:22
Срок аренды истекает	: 22 апреля 2018 г. 9:08:26
Основной шлюз	: fe80::1%13 192.168.100.1
DHCP-сервер	: 192.168.100.1
IAID DHCPv6	: 382489263
DUID клиента DHCPv6	: 00-01-00-01-15-80-FF-C7-98-4B-E1-C1-EE-95

Рис. 9.75. IAID интерфейса

Выбор IPv6-адресов DHCPv6-сервером

Сервер выбирает адреса, которые будут назначены соответствующей IA, на основе политики, определенной администратором, и комбинации следующей информации о клиенте:

- канале, к которому подключен клиент. Клиент может быть подключен к серверу напрямую или через relay-агента;
- идентификаторе DUID, предоставленном клиентом;
- информации из опций, предоставленной клиентом;
- информации из опций, предоставленной relay-агентом.

9.5.4. Stateful DHCPv6

Stateful DHCPv6 позволяет серверам назначать клиентам IPv6-адреса и другие конфигурационные параметры. Существуют два режима работы Stateful DHCPv6: обычный режим, при котором происходит обмен четырьмя сообщениями, и режим Rapid Commit, использующий два сообщения в процессе обмена.

В обычном режиме работы Stateful DHCPv6 клиент и сервер выполняют обмен четырьмя сообщениями (рис. 9.76), аналогично DHCPv4.

Source	Destination	Protocol	Info
fe80::cc7f:2dc1:4623:249a ff02::1:2		DHCPv6	Solicit XID: 0x2dd8d5 CID: 0001000116d51b2c206a8a72a582
fe80::86c9:b2ff:fe1f:9c01 fe80::cc7f:2dc1:4623:249a		DHCPv6	Advertise XID: 0x2dd8d5 CID: 0001000116d51b2c206a8a72a582 IAA: 201...
fe80::cc7f:2dc1:4623:249a ff02::1:2		DHCPv6	Request XID: 0x2dd8d5 CID: 0001000116d51b2c206a8a72a582 IAA: 2012:...
fe80::86c9:b2ff:fe1f:9c01 fe80::cc7f:2dc1:4623:249a		DHCPv6	Reply XID: 0x2dd8d5 CID: 0001000116d51b2c206a8a72a582 IAA: 2012::3...

Рис. 9.76. Обмен четырьмя сообщениями Stateful DHCPv6

Рассмотрим процесс обмена подробнее.

1. **Обнаружение DHCPv6.** После создания IA для своего интерфейса клиент начинает процесс обнаружения DHCP-серверов и запроса адресов и другой конфигурационной информации. Для этого он отправляет сообщение SOLICIT на адрес ff02::1:2 (*All_DHCP_Relay_Agents_and_Servers*). Оно включает номер транзакции; опцию *Client Identifier* для идентификации клиента сервером; опцию *Identity Association for Non-temporary Addresses* (*IA_NA*) для запроса постоянных адресов и/или опцию *Identity Association for Temporary Addresses* (*IA_TA*) для запроса временных адресов для соответствующего интерфейса; опцию *Option Request*, в которой указан набор конфигурационных параметров, которые клиент желает получить.

2. **Объявление DHCPv6.** В ответ на корректное сообщение SOLICIT (рис. 9.77) сервер отправляет сообщение ADVERTISE с предлагаемыми конфигурационными параметрами. Если сообщение SOLICIT было получено сервером от клиента напрямую, сообщение ADVERTISE отправляется на индивидуальный Link-Local IPv6-адрес клиента. Если сообщение SOLICIT было получено через relay-агента, сообщение ADVERTISE помещается как опция в сообщение RELAY-REPLY, которое отправляется на индивидуальный адрес relay-агента.

Сервер должен включить в сообщение ADVERTISE (рис. 9.78) скопированные из сообщения *Solicit* номер транзакции и опцию *Client Identifier*; опцию *Server Identifier* для идентификации себя; опции *IA_NA* и/или *IA_TA*, содержащие предлагаемые адреса. Сервер может включить в сообщение опцию *Preference*, которая содержит значение предпочтительности. Это значение конфигурируется администратором.

3. **Запрос DHCPv6.** Клиент должен игнорировать полученное сообщение ADVERTISE, в котором значение опции *Status Code* равно *NoAddrsAvail*.

При получении одного или нескольких корректных сообщений ADVERTISE от одного или нескольких серверов клиент выбирает лучшее из них на основе следующих критериев:

9. Протоколы уровня приложений

```
Internet Protocol Version 6, Src: fe80::cc7f:2dc1:4623:249a, Dst: ff02::1:2
User Datagram Protocol, Src Port: 546, Dst Port: 547
DHCPv6
  Message type: Solicit (1)
  Transaction ID: 0x2dd8d5
  Elapsed time
    Option: Elapsed time (8)
    Length: 2
    Value: 0000
    Elapsed time: 0ms
  Client Identifier
    Option: Client Identifier (1)
    Length: 14
    Value: 0001000116d51b2c206a8a72a582
    DUID: 0001000116d51b2c206a8a72a582
    DUID Type: link-layer address plus time (1)
    Hardware type: Ethernet (1)
    DUID Time: Feb 20, 2012 17:54:36.000000000 RTZ 2 (зима)
    Link-layer address: 20:6a:8a:72:a5:82
  Identity Association for Non-temporary Address
    Option: Identity Association for Non-temporary Address (3)
    Length: 12
    Value: 11206a8a0000000000000000
    IAID: 11206a8a
    T1: 0
    T2: 0
  Fully Qualified Domain Name
  Vendor Class
  Option Request
    Option: Option Request (6)
    Length: 8
    Value: 0018001700110027
    Requested Option code: Domain Search List (24)
    Requested Option code: DNS recursive name server (23)
    Requested Option code: Vendor-specific Information (17)
    Requested Option code: Fully Qualified Domain Name (39)
```

Рис. 9.77. Сообщение SOLICIT

- предпочтение отдается сообщению ADVERTISE с наивысшим значением предпочтительности;
- если несколько сообщений ADVERTISE имеют одинаковое значение предпочтительности, клиент выбирает тот сервер, который предлагает наиболее интересные клиенту конфигурационные параметры;
- клиент может выбрать сообщение ADVERTISE с наименьшим значением предпочтительности, если сервер предлагает наилучший набор объявляемых параметров.

```
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c01, Dst: fe80::cc7f:2dc1:4623:249a
User Datagram Protocol, Src Port: 547, Dst Port: 546
DHCPv6
  Message type: Advertise (2)
  Transaction ID: 0x2dd8d5
  Client Identifier
  Server Identifier
    Option: Server Identifier (2)
    Length: 10
    Value: 0003000684c9b21f9c00
    DUID: 0003000684c9b21f9c00
    DUID Type: link-layer address (3)
    Hardware type: IEEE 802 (6)
    Link-layer address: 84:c9:b2:1f:9c:00
  Identity Association for Non-temporary Address
    Option: Identity Association for Non-temporary Address (3)
    Length: 40
    Value: 11206a8a00049d400007620000050018201200000000000...
    IAID: 11206a8a
    T1: 302400
    T2: 483840
  IA Address
    Option: IA Address (5)
    Length: 24
    Value: 20120000000000000000000000000000342100093a8000278d00
    IPv6 address: 2012::3421
    Preferred lifetime: 604800
    Valid lifetime: 2592000
  Preference
```

Рис. 9.78. Сообщение ADVERTISE

Как только клиент выбрал сообщение ADVERTISE, он сохраняет информацию о сервере (значение предпочтительности, объявленный адрес, время получения объявления и т. д.). Далее для запроса предложенных конфигурационных параметров он отправляет на адрес ff02::1:2 сообщение REQUEST (рис. 9.79). Сообщение REQUEST должно включать опцию *Server Identifier*, которая содержит идентификатор выбранного сервера; опцию *Client Identifier* для идентификации клиента и опцию *IA*, содержащую предложенные для соответствующего интерфейса IPv6-адреса. Отправка сообщения на групповой адрес гарантирует, что все серверы, приславшие предложения, будут проинформированы о выборе клиента. Если необходимо, это сообщение будет перенаправлено relay-агентом соответствующим серверам. Серверы, которые не были выбраны клиентом, могут отменить резервирование предложенных IPv6-адресов.

4. Ответ DHCPv6. Когда сервер получает корректное сообщение REQUEST, он в соответствии с настроенной политикой создает для этого клиента привязку (binding) (рис. 9.80), где сохраняет назначеннюю конфигурационную информацию.

Сервер отправляет клиенту одноадресное сообщение REPLY, содержащее конфигурационные параметры, аналогичные ранее предложенными.

9. Протоколы уровня приложений

```
Internet Protocol Version 6, Src: fe80::cc7f:2dc1:4623:249a, Dst: ff02::1:2
User Datagram Protocol, Src Port: 546, Dst Port: 547
DHCPv6
  Message type: Request (3)
  Transaction ID: 0x2dd8d5
  Elapsed time
  Client Identifier
  Server Identifier
  Identity Association for Non-temporary Address
    Option: Identity Association for Non-temporary Address (3)
    Length: 40
    Value: 11206a8a00049d40000762000005001820120000000000...
    IAID: 11206a8a
    T1: 302400
    T2: 483840
  IA Address
    Option: IA Address (5)
    Length: 24
    Value: 20120000000000000000000000000000342100093a8000278d00
    IPv6 address: 2012::3421
    Preferred lifetime: 604800
    Valid lifetime: 2592000
  Fully Qualified Domain Name
  Vendor Class
  Option Request
    Option: Option Request (6)
    Length: 8
    Value: 0018001700110027
    Requested Option code: Domain Search List (24)
    Requested Option code: DNS recursive name server (23)
    Requested Option code: Vendor-specific Information (17)
    Requested Option code: Fully Qualified Domain Name (39)
```

Рис. 9.79. Сообщение REQUEST

```
Switch(config)#sh ipv6 dhcp binding
Client DUID : 0001000116d51b2c206a8a72a582
               address: 2001:DB8::4
                           preferred lifetime 100 ,valid lifetime 200

Total Entries: 1
```

Рис. 9.80. Привязка, созданная сервером для клиента

В сообщение вклюаются опции, идентифицирующие сервер и клиента, а также опции *IA* с назначенными адресами и другими конфигурационными параметрами.

Если сервер обнаруживает, что в опции *IA* сообщения клиента префикс одного или нескольких IPv6-адресов не соответствует каналу подключения, в сообщение REPLY (рис. 9.81) помещается опция *Status Code* со значением *NoOnLink*.

Если сервер не может назначить адрес интерфейсу клиента, опция *IA* сообщения REPLY не содержит адреса, а в опции *Status Code* устанавливается значение *NoAddrsAvail*.

```
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c01, Dst: fe80::cc7f:2dc1:4623:249a
User Datagram Protocol, Src Port: 547, Dst Port: 546
DHCPv6
  Message type: Reply (7)
  Transaction ID: 0x2dd8d5
  Client Identifier
    Option: Client Identifier (1)
    Length: 14
    Value: 0001000116d51b2c206a8a72a582
    DUID: 0001000116d51b2c206a8a72a582
    DUID Type: link-layer address plus time (1)
    Hardware type: Ethernet (1)
    DUID Time: Feb 20, 2012 17:54:36.000000000 RTZ 2 (зима)
    Link-layer address: 20:6a:8a:72:a5:82
  Server Identifier
    Option: Server Identifier (2)
    Length: 10
    Value: 0003000684c9b21f9c00
    DUID: 0003000684c9b21f9c00
    DUID Type: link-layer address (3)
    Hardware type: IEEE 802 (6)
    Link-layer address: 84:c9:b2:1f:9c:00
  Identity Association for Non-temporary Address
    Option: Identity Association for Non-temporary Address (3)
    Length: 40
    Value: 11206a8a00049d400007620000050018201200000000000...
    IAID: 11206a8a
    T1: 302400
    T2: 483840
  IA Address
    Option: IA Address (5)
    Length: 24
    Value: 20120000000000000000000000000000342100093a8000278d00
    IPv6 address: 2012::3421
    Preferred lifetime: 604800
    Valid lifetime: 2592000
```

Рис. 9.81. Сообщение REPLY

Когда клиент получает сообщение REPLY, он извлекает из него конфигурационную информацию и проверяет опцию *Status Code*. Прежде чем применить конфигурационные параметры, клиент должен выполнить проверку дублирования адресов. Если адрес не уникален, клиент отправляет серверу

сообщение DECLINE, чтобы сообщить, что он уже используется на канале. Если адрес уникален, клиент обновляет параметры соответствующего интерфейса и запоминает длительность таймеров T1 и T2, указанных в опции *Identity Association for Non-temporary Addresses (IA_NA)*. Таймер T1 показывает время, через которое клиент должен отправить сообщение RENEW серверу, который первоначально предоставил ему адрес и конфигурационные параметры для продления аренды адреса и обновления конфигурационных параметров. Таймер T2 показывает время, через которое клиент должен отправить сообщение REBIND любому доступному серверу, чтобы продлить время аренды адреса и других конфигурационных параметров. Это сообщение отправляется, если не пришел ответ от сервера на сообщение RENEW.

Если клиент получил сообщение REPLY со значением NotOnLink в опции *Status Code*, он или повторно отправляет сообщение REQUEST без указания адресов, или перезапускает процесс обнаружения серверов.

Если сообщение REPLY содержит опцию *Status Code* со значением NoAddrsAvail, клиент или перезапускает процесс обнаружения серверов, или отправляет сообщение INFORMATION-REQUEST, чтобы получить только конфигурационные параметры.

Режим Rapid Commit

В обычном режиме работы Stateful DHCPv6 клиент получает адреса и конфигурационную информацию в результате обмена четырьмя сообщениями: SOLICIT, ADVERTISE, REQUEST и REPLY. Для ускорения конфигурирования клиента в сети с высокой загрузкой может использоваться режим Rapid Commit. В этом режиме клиент и сервер обмениваются всего двумя сообщениями: SOLICIT и REPLY. Для активизации этого режима и клиент, и сервер должны поддерживать опцию *Rapid Commit*. Она отдельно конфигурируется на клиенте и на сервере.

Когда на клиенте настроена опция *Rapid Commit*, он включает ее в сообщение SOLICIT (рис. 9.82).

Если сервер, который настроен для обработки опции *Rapid Commit*, получил от клиента содержащее ее сообщение SOLICIT, он отправляет в ответ сообщение REPLY. Прежде чем отправить сообщение REPLY, сервер назначает адреса и другие конфигурационные параметры. В сообщение REPLY сервер включает опцию *Rapid Commit* (рис. 9.83), чтобы показать, что это ответ на сообщение SOLICIT.

При получении сообщения REPLY клиент не отправляет сообщение REQUEST, так как подразумевается, что адреса ему уже выделены, а не предложены.

Обычно в сети для обработки опции *Rapid Commit* настраивается только один сервер. Если клиенту ответят несколько серверов, он будет использовать адреса, назначенные только одним из них.

Если сервер не поддерживает опцию *Rapid Commit*, в ответ на сообщение SOLICIT он отправит сообщение ADVERTISE (т. е. продолжит работу в обычном режиме).

Технологии TCP/IP в современных компьютерных сетях

```
Internet Protocol Version 6, Src: fe80::aef1:ffff:feb5:fc00, Dst: ff02::1:2
User Datagram Protocol, Src Port: 546, Dst Port: 547
DHCPv6
  - Message type: Solicit (1)
  - Transaction ID: 0x005956
  - Client Identifier
    - Option: Client Identifier (1)
    - Length: 10
    - Value: 00030006acf1dfb5fc00
    - DUID: 00030006acf1dfb5fc00
    - DUID Type: link-layer address (3)
    - Hardware type: IEEE 802 (6)
    - Link-layer address: ac:f1:df:b5:fc:00
  - Identity Association for Non-temporary Address
    - Option: Identity Association for Non-temporary Address (3)
    - Length: 12
    - Value: 000000010000000000000000
    - IAID: 00000001
    - T1: 0
    - T2: 0
  - Rapid Commit
    - Option: Rapid Commit (14)
    - Length: 0
  - Elapsed time
```

Рис. 9.82. Сообщение SOLICIT с опцией Rapid Commit

```
Internet Protocol Version 6, Src: fe80::12be:f5ff:fedc:9600, Dst: fe80::aef1:ffff:feb5:fc00
User Datagram Protocol, Src Port: 547, Dst Port: 546
DHCPv6
  - Message type: Reply (7)
  - Transaction ID: 0x005956
  - Client Identifier
  - Server Identifier
  - Identity Association for Non-temporary Address
    - Option: Identity Association for Non-temporary Address (3)
    - Length: 40
    - Value: 00000001000000320000050005001820010db8000000...
    - IAID: 00000001
    - T1: 50
    - T2: 80
  - IA Address
    - Option: IA Address (5)
    - Length: 24
    - Value: 20010db8000000000000000000000000020000064000000c8
    - IPv6 address: 2001:db8::2
    - Preferred lifetime: 100
    - Valid lifetime: 200
  - Rapid Commit
    - Option: Rapid Commit (14)
    - Length: 0
  - Preference
```

Рис. 9.83. Сообщение REPLY с опцией Rapid Commit

9. Протоколы уровня приложений

Рассмотрим примеры настройки Stateful DHCPv6 в обоих режимах.

Пример 1. В первом примере клиенты могут получить префикс IPv6-адреса и адрес DNS-сервера, используя обычный режим Stateful DHCPv6 (рис. 9.84). DHCPv6-сервер настроен на коммутаторе. Для одного пула DHCPv6 (IPv6 DHCP pool) может быть сконфигурирован только один префикс. Пул должен быть привязан к соответствующему интерфейсу. Когда коммутатор принимает от клиента запрос на получение адреса, он проверяет DHCP pool, ассоциированный с входным интерфейсом и назначает соответствующий префикс.

Следует отметить, что интерфейс VLAN по умолчанию (VLAN 1) применяется ко всем портам коммутатора.

Настройка DHCPv6-сервера на коммутаторе DGS-3630-28TC

- Активировать DHCPv6-сервер глобально на коммутаторе:

```
service ipv6 dhcp
```

- Создать пул IPv6 DHCP:

```
ipv6 dhcp pool serv_pl6
```

```
address prefix 2001:DB8::0/64 lifetime 200 100
```

```
dns-server 2001:DB8:3000::42
```

```
domain-name pl6.com
```

- Ассоциировать пул serv_pl6 с VLAN по умолчанию:

```
interface vlan 1
```

```
ipv6 dhcp server serv_pl6
```

```
ipv6 address 2001:DB8::1/64
```

DHCP-сервер

IPv6 DHCP pool:

префикс 2001:DB8::0/64

DNS-сервер 2001:DB8:3000::42

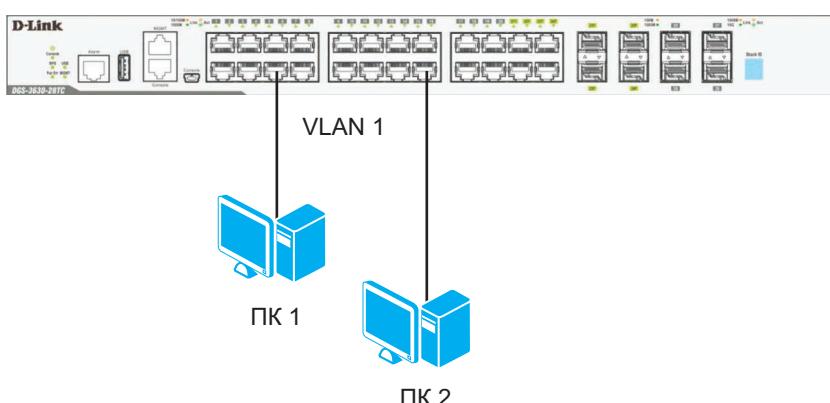


Рис. 9.84. Схема сети

Пример 2. Во втором примере клиент получит префикс IPv6-адреса и адрес DNS-сервера, используя режим Rapid Commit. На клиенте и сервере DHCPv6 должна быть настроена поддержка опции *Rapid Commit* (рис. 9.85).

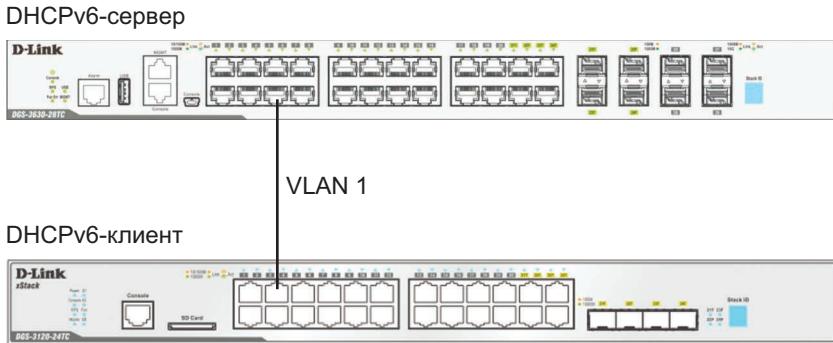


Рис. 9.85. Схема сети

Настройка DHCPv6-сервера на коммутаторе DGS-3630-28TC

- Активировать DHCPv6-сервер глобально на коммутаторе:
service ipv6 dhcp

- Создать пул IPv6 DHCP:

```
ipv6 dhcp pool serv_pl6
address prefix 2001:DB8::0/64 lifetime 200 100
dns-server 2001:DB8:3000::42
domain-name pl6.com
```

- Ассоциировать пул *serv_pl6* с *VLAN* по умолчанию:

```
interface vlan 1
ipv6 dhcp server serv_pl6 rapid-commit
ipv6 address 2001:DB8::1/64
```

Настройка DHCPv6-клиента на коммутаторе DGS-3120-24TC

- Настроить автоматическое получение IPv6-адреса на интерфейсе System:
config ipif System dhcp

- Включить DHCPv6-клиент с поддержкой опции Rapid commit:

```
config ipif System dhcpcv6_client enable rapid_commit
```

9.5.5. Stateless DHCPv6

Stateless DHCPv6 описан в DFC 3736. Он служит для получения дополнительных конфигурационных параметров, таких как адреса DNS-, SIP-серверов, и не выполняет отслеживание состояния клиентов. Узел, который использует Stateless DHCPv6, должен получить IPv6-адрес через другой механизм — вручную или через stateless autoconfiguration.

9. Протоколы уровня приложений

Клиент сообщает, что он запрашивает конфигурационную информацию, отправляя сообщение INFORMATION-REQUEST, которое включает опцию *Option Request* (рис. 9.86). В ней перечисляются все опции, которые клиент желает получить от DHCPv6-сервера. Опция *IA* в сообщении INFORMATION-REQUEST отсутствует.

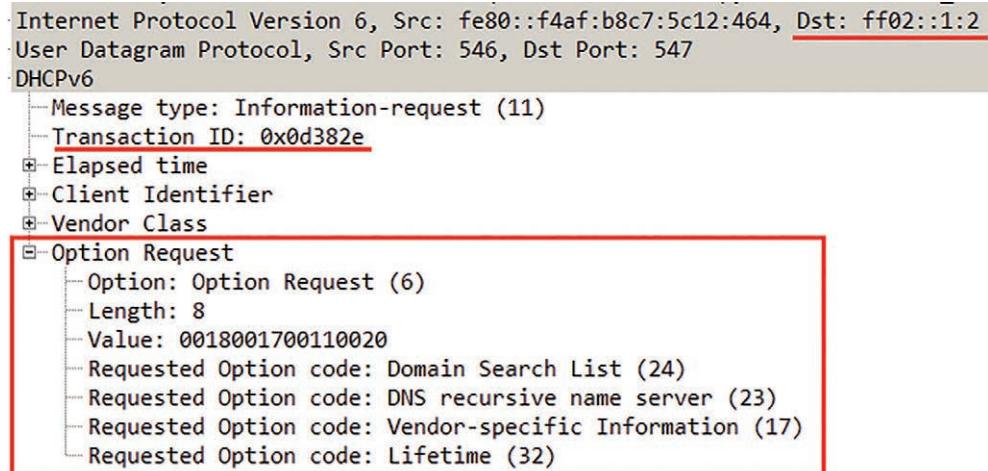


Рис. 9.86. Сообщение INFORMATION-REQUEST

При получении сообщения INFORMATION-REQUEST сервер определяет соответствующие конфигурационные параметры для клиента и отправляет сообщение REPLY (рис. 9.87), содержащее их.

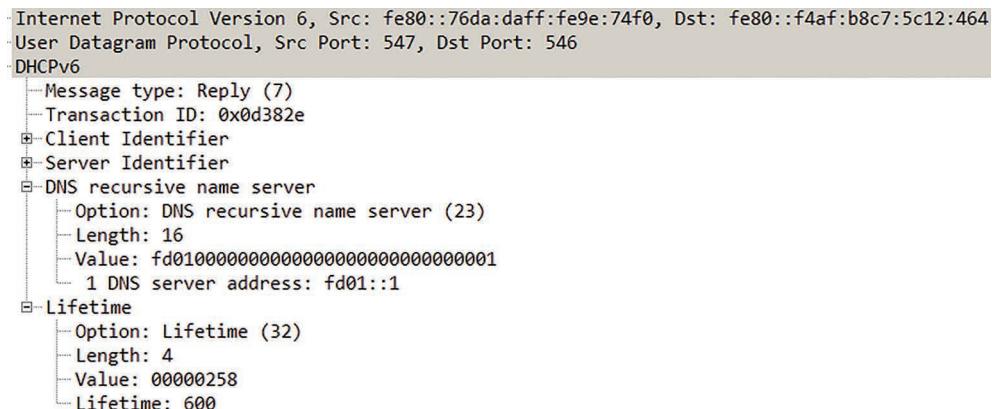


Рис. 9.87. Сообщение REPLY

Если клиент подключен к серверу через relay-агента, то Stateless DHCPv6 дополнительно использует следующие сообщения:

- Relay-forward — отправляется relay-агентом и содержит сообщение клиента, предназначенное серверу;
- Relay-reply — отправляется DHCPv6-сервером relay-агенту и содержит сообщение, предназначенное клиенту.

9.5.6. DHCPv6 Prefix Delegation

Опция *Prefix Delegation* предоставляет механизм автоматического делегирования префиксов IPv6 для сетей с помощью протокола DHCPv6 (рис. 9.88). Он описан в RFC 3633 и 3769. Механизм делегирования префикса позволяет централизованно контролировать назначение IPv6-префиксов сетям и избежать их потенциальных конфликтов. Его можно использовать провайдерам услуг при назначении префикса сетям IPv6 клиентов или в корпоративных сетях для назначения префиксов IPv6 сетям удаленных офисов.

Делегирование префикса IPv6 выполняет делегирующий маршрутизатор при получении запроса от запрашивающего маршрутизатора. *Делегирующий маршрутизатор* (delegating router) — это маршрутизатор, работающий как DHCP-сервер и отвечающий на запрос префикса. *Запрашивающий маршрутизатор* (requesting router) — это маршрутизатор, работающий как DHCP-клиент и запрашивающий назначение префикса.

Делегирующему маршрутизатору не известна топология сети, к которой подключен запрашивающий маршрутизатор, и он не требует предоставления информации о ней. При выборе префикса для делегирования ему необходим только идентификатор запрашивающего маршрутизатора.

После получения префикса запрашивающий маршрутизатор использует его для назначения IPv6-адресов устройствам, подключенными к его локальным интерфейсам.

Поскольку делегирование префикса реализуется с помощью опции, то между запрашивающим и делегирующим маршрутизатором происходит обмен теми же сообщениями, которые описаны в RFC 3315: SOLICIT, ADVERTISE, REQUEST и REPLY. В режиме Rapid Commit они обмениваются двумя сообщениями: SOLICIT и REPLY.

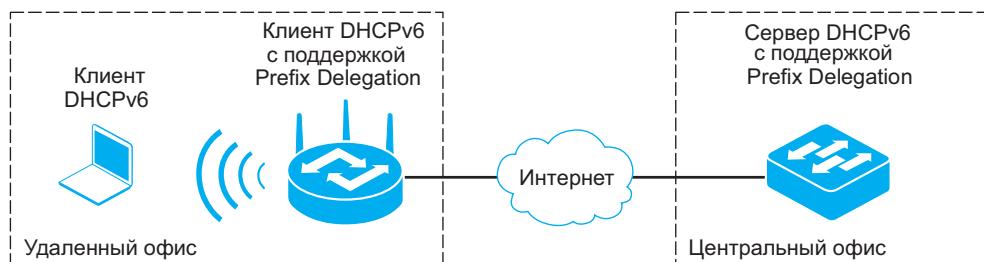
Процессы делегирования префикса и назначения адресов IPv6 через DHCPv6 независимы друг от друга. Запрашивающий маршрутизатор может использовать DHCPv6 или только для делегирования префикса, или для делегирования префикса наряду с назначением IPv6-адресов и другой конфигурационной информации.

Запрашивающий маршрутизатор создает *Identity Association for Prefix Delegation* (*IA_PD*). Идентификация идентичности для делегирования префикса (*IA_PD*) — это структура, с помощью которой делегирующий маршрутизатор и запрашивающий маршрутизатор могут идентифицировать, группировать и управлять набором префиксов IPv6. Каждая *IA_PD* состоит из идентификатора *IAID* и конфигурационной информации.

9. Протоколы уровня приложений

В отличие от *IA*, которая идентифицирует определенный интерфейс, *IA_PD* может быть ассоциирована с запрашивающим маршрутизатором, с набором интерфейсов или конкретным интерфейсом. Запрашивающий маршрутизатор должен создать хотя бы одну *IA_PD*. Он может ассоциировать ее с каждым из своих внутренних (локальных) интерфейсов и использовать для получения префикса от делегирующего маршрутизатора.

Использование Prefix Delegation в корпоративной сети



Использование Prefix Delegation в сети провайдера услуг

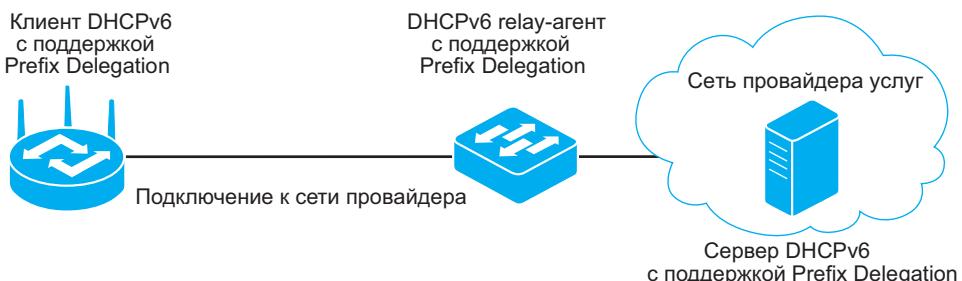


Рис. 9.88. Использование метода делегирования префикса в сетях

IA_PD идентифицируется *IAID*. Запрашивающий маршрутизатор должен выбирать уникальный *IAID* для каждой созданной им *IA_PD*. Конфигурационная информация *IA_PD* состоит из одного или более префиксов IPv6, назначенных запрашивающему маршрутизатору.

Работает DHCPv6 Prefix Delegation при обмене четырьмя сообщениями следующим образом.

1. Запрашивающий маршрутизатор создает *IA_PD* и присваивает ей идентификатор *IAID*. Далее он отправляет сообщение *Solicit*, содержащее опцию *IA_PD* (рис. 9.89).

2. Если делегирующий маршрутизатор может предложить префиксы, он включает их в опцию *IA_PD* сообщения *ADVERTISE* (рис. 9.90).

3. Когда запрашивающий маршрутизатор выбрал делегирующий маршрутизатор, он отправляет ему запрос REQUEST с опцией *IA_PD* (рис. 9.91).

4. Делегирующий маршрутизатор возвращает назначенные префиксы в опции *IA_PD* сообщения REPLY (рис. 9.92).

```
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c00, Dst: ff02::1:2
User Datagram Protocol, Src Port: 546, Dst Port: 547
DHCPv6
  Message type: Solicit (1)
  Transaction ID: 0x003cbb
  Client Identifier
  Elapsed time
  Option Request
    Identity Association for Prefix Delegation
      Option: Identity Association for Prefix Delegation (25)
      Length: 12
      Value: 000000010000000000000000
      IAID: 00000001
      T1: 0
      T2: 0
```

Рис. 9.89. Сообщение SOLICIT с опцией *Identity Association for Prefix Delegation*

```
Internet Protocol Version 6, Src: fe80::12be:f5ff:fedc:9600, Dst: fe80::86c9:b2ff:fe1f:9c00
User Datagram Protocol, Src Port: 547, Dst Port: 546
DHCPv6
  Message type: Advertise (2)
  Transaction ID: 0x003cbb
  Client Identifier
  Server Identifier
  Preference
    Identity Association for Prefix Delegation
      Option: Identity Association for Prefix Delegation (25)
      Length: 41
      Value: 000000010000003200000050001a001900000064000000c8...
      IAID: 00000001
      T1: 50
      T2: 80
    IA Prefix
      Option: IA Prefix (26)
      Length: 25
      Value: 0000064000000c84030040db80000000000000000000000...
      Preferred lifetime: 100
      Valid lifetime: 200
      Prefix length: 64
      Prefix address: 3004:db8::
```

Рис. 9.90. Сообщение ADVERTISE с опцией *Identity Association for Prefix Delegation*

9. Протоколы уровня приложений

```
Internet Protocol Version 6, Src: fe80::86c9:b2ff:fe1f:9c00, Dst: ff02::1:2
User Datagram Protocol, Src Port: 546, Dst Port: 547
DHCPv6
  Message type: Request (3)
  Transaction ID: 0x001379
  Client Identifier
  Server Identifier
  Elapsed time
  Option Request
  Identity Association for Prefix Delegation
    Option: Identity Association for Prefix Delegation (25)
    Length: 41
    Value: 00000010000000000000000000000001a001900000064000000c8...
    IAID: 00000001
    T1: 0
    T2: 0
  IA Prefix
    Option: IA Prefix (26)
    Length: 25
    Value: 00000064000000c84030040db800000000000000000000000...
    Preferred lifetime: 100
    Valid lifetime: 200
    Prefix length: 64
    Prefix address: 3004:db8::
```

Рис. 9.91. Сообщение REQUEST с опцией *Identity Association for Prefix Delegation*

Рассмотрим пример настройки делегирования префикса. Предположим, что клиент, подключенный в Интернет посредством IPoE, желает, чтобы провайдер обеспечил делегирование префикса. Он будет использоваться для назначения IPv6-адресов устройствам локальной сети. Пусть DHCPv6-сервер с поддержкой Prefix Delegation (PD) настроен на пограничном коммутаторе провайдера. Клиентом DHCPv6 с поддержкой Prefix Delegation является абонентский маршрутизатор. Для получения префикса абонентский маршрутизатор должен быть подключен к сети провайдера через интерфейс WAN. Настройка приведена для коммутатора модели DGS-3630-28TC и маршрутизатора модели DIR-825 (рис. 9.93).

Настройка DGS-3630-28TC

- Активировать DHCPv6-сервер глобально на коммутаторе:
service ipv6 dhcp
- Создать пул IPv6 для делегирования префикса:
ipv6 local pool pd_vlan1 3004:DB8::/48 64
ipv6 dhcp pool pool1
prefix-delegation pool pd_vlan1 lifetime 300 200

- Ассоциировать пул pool1 с VLAN по умолчанию:

```
interface vlan 1
```

```
ipv6 dhcp server pool1
```

```
ipv6 address 2001:DB8::1/64
```

```
Internet Protocol Version 6, Src: fe80::12be:f5ff:fedc:9600, Dst: fe80::86c9:b2ff:fe1f:9c00
User Datagram Protocol, Src Port: 547, Dst Port: 546
DHCPv6
  - Message type: Reply (7)
  - Transaction ID: 0x001379
  - Client Identifier
  - Server Identifier
  - Identity Association for Prefix Delegation
    - Option: Identity Association for Prefix Delegation (25)
    - Length: 41
    - Value: 000000010000003200000050001a001900000064000000c8...
    - IAID: 00000001
    - T1: 50
    - T2: 80
  - IA Prefix
    - Option: IA Prefix (26)
    - Length: 25
    - Value: 00000064000000c84030040db800000000000000000000000...
    - Preferred lifetime: 100
    - Valid lifetime: 200
    - Prefix length: 64
    - Prefix address: 3004:db8::
```

Рис. 9.92. Сообщение REPLY с опцией *Identity Association for Prefix Delegation*

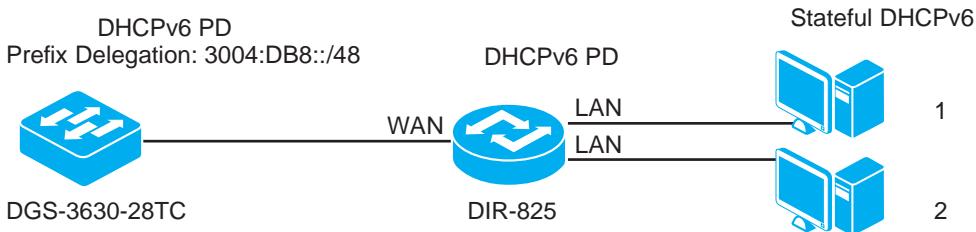


Рис. 9.93. Схема сети

Настройка DIR-825

- Настроить на WAN-интерфейсе соединение типа «Динамический IPv6» (рис. 9.94):

Настройка соединения → *WAN* → *Добавить* → *Динамический IPv6*. В выпадающем меню *Получить IPv6* выбрать через *DHCPv6 PD*.

- Настроить режим динамического назначения IPv6-адресов на LAN-интерфейсах (рис. 9.95):

Настройка соединения → *LAN* → *IPv6*. В разделе *Локальный IPv6* выбрать *Делегирование префикса*. В разделе *Динамический IPv6* выбрать *Stateful*, в поле *Начальный IPv6* ввести *3004:db8::2*, в поле *Конечный IPv6* — *3004:db8::5*.

9. Протоколы уровня приложений

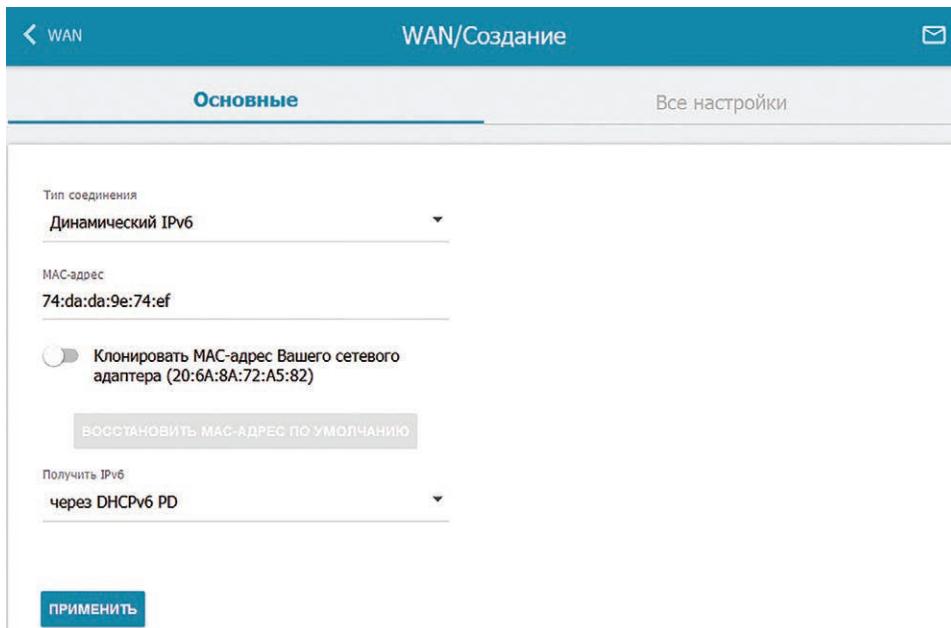


Рис. 9.94. Настройка типа соединения Динамический IPv6

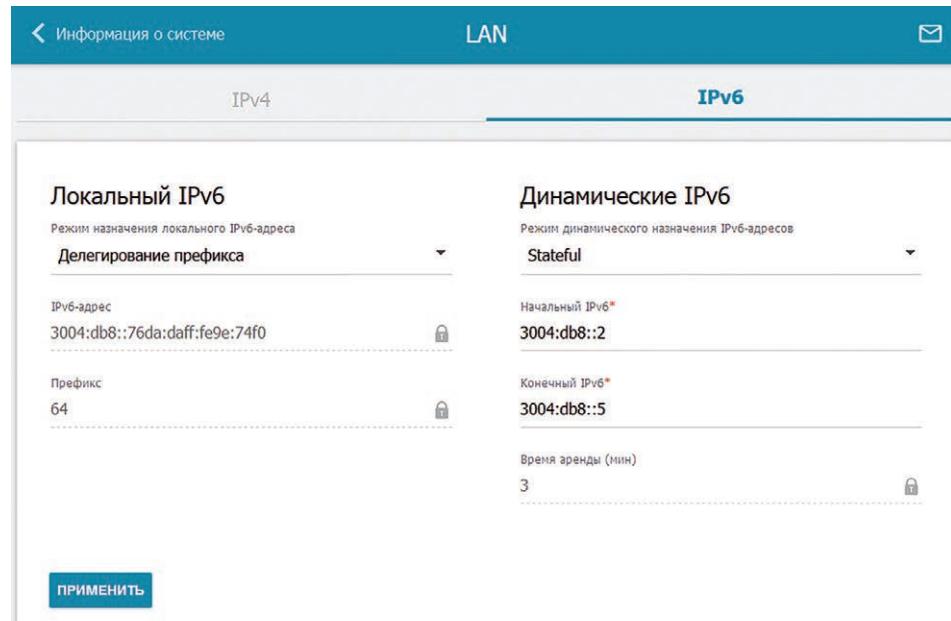


Рис. 9.95. Настройка динамического назначения IPv6-адресов для устройств локальной сети

9.5.7. Опции DHCPv6 Relay Agent

DHCPv6 relay-агент может добавлять дополнительную информацию в сообщения DHCPv6, передаваемые DHCPv6-серверу. Эта информация аналогична информации опции 82, поддерживаемой relay-агентом DHCPv4.

DHCPv6 relay-агент поддерживает следующие опции:

- *Interface-id* (option 18): используется relay-агентом DHCPv6 для идентификации интерфейса, на который было получено сообщение от клиента. Аналогична подопции *Agent Circuit ID* опции 82 протокола DHCPv4.

- *Relay Agent Remote-ID* (option 37): строка, идентифицирующая relay-агента. Аналогична подопции *Agent Remote ID* опции 82 протокола DHCPv4.

Чтобы использовать опции 18 и 37 в сети, их поддержка должна быть реализована и на DHCPv6 relay-агенте, и на DHCPv6-сервере.

Relay-агент добавляет опции 18 и 37 в сообщение RELAY-FORW. DHCPv6-сервер использует полученную информацию для выбора определенных IPv6-адресов, префиксов и других параметров для конкретных узлов. Сервер может не повторять информацию опций 18 и 37 в сообщении RELAY-REPL. Если информация была добавлена, relay-агент будет удалять опции 18 и 37 перед пересылкой DHCPv6-сообщения клиенту.

Relay-агенты с поддержкой опций 18 и 37 могут выполнять следующие политики перенаправления DHCPv6-сообщений серверу:

1. **Drop**: политика отбрасывания DHCPv6-сообщений, которые уже содержат опцию 18 или 37.

2. **Keep**: политика сохранения опции 18 или 37 в полученном DHCv6-сообщении. Если relay-агент получает DHCPv6-сообщение, содержащее опцию 18 или 37, оно будет без изменений перенаправлено DHCPv6-серверу. Если DHCPv6-сообщение не содержит опцию 18 или 37, она будет в него добавлена при пересылке DHCPv6-серверу.

9.5.8. Функция DHCPv6 Guard

Функция DHCPv6 Guard (рис. 9.96) позволяет портам коммутатора блокировать сообщения Advertise и REPLY, которые приходят от неавторизованных DHCPv6-серверов и relay-агентов, перенаправляющих сообщения от серверов. Сообщения от клиентов не блокируются. Функция определяет роли подключенных устройств: клиент или сервер. Если подключенному к коммутатору устройству назначена роль клиента (роль по умолчанию), порт будет блокировать все серверные сообщения (Advertise и REPLY). Эти сообщения будут приниматься портом коммутатора в том случае, если подключенному устройству назначена роль сервера. Если не настроены списки управления доступом (ACL) по IPv6-адресам источника, коммутатор будет передавать все сообщения Advertise и REPLY. Если ACL настроены, сообщения Advertise и REPLY, в которых указаны недостоверные IPv6-адреса DHCPv6-серверов, будут отбрасываться.

9. Протоколы уровня приложений

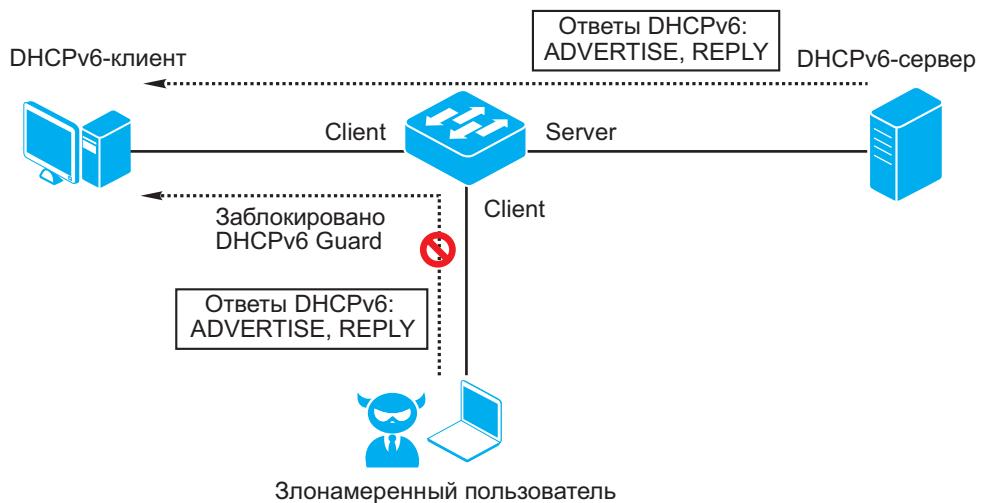


Рис. 9.96. Функция DHCPv6 Guard

10. Поиск неисправностей в сетях TCP/IP

Поиск неисправностей в компьютерных сетях является одной из задач системных администраторов и сотрудников технической поддержки. Сети состоят из множества элементов: компьютеров, принтеров, IP-телефонов, серверов, систем хранения информации, сетевого оборудования, различного рода сетевого программного обеспечения и т. д. Устройства подключаются друг к другу с помощью кабелей или по беспроводной связи.

Из-за общего усложнения сетевой среды возрастает вероятность появления проблем со связью и снижения производительности. Администратор сети должен владеть методиками анализа сетевых проблем с точки зрения различных уровней модели TCP/IP или OSI и уметь использовать необходимые средства для эффективного выявления и устранения проблем со связью.

Первым шагом в решении сетевой проблемы является обнаружение ее источника. Сетевые проблемы могут произойти в той части сети, которую контролирует администратор, а могут быть частью внешней сети (сети провайдера). В этом разделе будут рассматриваться методики анализа сетевых проблем в локальных сетях.

10.1. Методика поиска неисправностей

Процесс поиска неисправностей в сетевой среде должен быть систематизирован. Необходимо определить конкретные признаки неисправности и выявить все потенциальные проблемы, которыми они могут быть вызваны. Затем устраниТЬ все возможные причины проблем и удостовериться, что все признаки неисправности исчезли.

Ниже приведены шаги процесса поиска неисправностей.

1. Получение подробной информации о возникшей проблеме. При диагностике любой проблемы полезно применять логический подход. Обычно при поиске неполадок следует искать ответы на такие вопросы:

- Что работает?
- Что не работает?
- Как между собой связаны вещи, которые работают и которые не работают?
- Работало ли когда-нибудь то, что сейчас не работает?
- Если да, то что изменилось с тех пор, когда оно работало?

Соберите все факты, которые позволяют наиболее точно установить возможные причины проблемы.

2. Воспроизведение проблемы при возможности. При возможности самостоятельно воспроизведите проблему. Это позволит выяснить, правильно ли вы поняли ее причину.

3. Локализация и изоляция причины. Попытайтесь определить место, откуда начать поиск неполадок и изолировать ошибку в устройстве, программном обеспечении или конфигурации, которые вызывают проблему.

4. Разработка плана устранения неисправностей в сети для решения проблемы. Исследуйте и/или рассмотрите возможные пути решения проблемы. Учитывайте возможность того, что некоторые доступные решения проблемы могут вызвать другие проблемы.

5. Реализация плана устранения неисправностей в сети. Фактическим решением проблемы может оказаться замена оборудования, кабеля, применение патча программного обеспечения, переустановка приложения или компонента либо очистка файла, зараженного вирусом. Если проблема заключается в учетной записи пользователя, может потребоваться изменение параметров безопасности пользователя или сценариев входа в систему. При некорректной настройке устройства потребуется исправление его конфигурации.

6. Проверка решения проблемы. После того, как было реализовано принятое решение, убедитесь, что проблема полностью решена.

7. Документирование проблемы и решения. Документация может использоваться в дальнейшем для помощи в устранении таких же или аналогичных проблем. Помимо того, можно использовать документацию для подготовки отчетов по общим проблемам сети для руководства и/или пользователей или для обучения новых пользователей сети либо сотрудников технической поддержки.

10.2. Средства поиска и устранения неполадок

Существует ряд утилит для настройки, администрирования и диагностики, которые можно использовать для устранения проблем TCP/IP. Эти средства перечислены в табл. 10.1.

Таблица 10.1. Утилиты для диагностики TCP/IP в ОС Windows

Утилита	Описание
<i>Arp</i>	Просмотр и редактирование кеша протокола ARP
<i>Hostname</i>	Отображение имени компьютера
<i>Ipconfig</i>	Просмотр текущей конфигурации TCP/IP для IPv4 и IPv6. Также используется для управления конфигурациями IP-адресов, распределенных с помощью протокола DHCP, просмотра или очистки кеша разрешения клиентов DNS и регистрации имен DNS
<i>Netsh</i>	Средство настройки многих сетевых служб. Для каждой сетевой службы имеется контекст с командами, специфичными для данной службы. Для контекстов <i>netsh interface ip</i> и <i>netsh interface ipv6</i> средство позволяет просматривать и управлять настройками протокола TCP/IP на локальном или удаленном компьютере
<i>Netstat</i>	Просмотр статистики протокола и сведений о текущих TCP-подключениях
<i>Nslookup</i>	Выполнение DNS-запросов и просмотр результатов

Утилита	Описание
<i>Ping</i>	Отправка эхо-запросов ICMP или ICMPv6 для проверки достоверности
<i>Route</i>	Просмотр таблиц маршрутизации IPv4 и IPv6 и редактирование таблицы маршрутизации IPv4
<i>Tracert</i>	Отправка эхо-запросов ICMP или ICMPv6 для прослеживания сетевого маршрута прохождения пакетов IPv4 или IPv6 к определенному объекту назначения
<i>Pathping</i>	Отправка эхо-запросов ICMP или ICMPv6 для прослеживания маршрута прохождения пакетов IPv4 или IPv6 к определенному объекту назначения и просмотр сведений о потерях пакетов по каждому маршрутизатору и связи на маршруте
<i>Telnet</i>	Проверка установки TCP-соединения между двумя узлами

Утилиты *ping* и *tracert/traceroute* наиболее полезны в диагностике IP-соединения. В большинстве операционных систем и сетевых устройств они установлены по умолчанию.

Также полезны при диагностике сетей аппаратные и программные анализаторы сетевого трафика (например, программный анализатор Wireshark) и программное обеспечение для мониторинга и управления по SNMP (например, D-Link D-View 7).

В коммутаторах D-Link с интерфейсом командной строки (CLI) для контроля параметров сети и поиска неисправностей используются команды **show**.

10.3. Анализ неисправностей

В любой сетевой среде должна быть в наличии своевременная и точная информация о сети, к которой может в любое время обратиться технический персонал. Только при наличии полной информации можно принять обоснованное решение о том, какие изменения должны быть сделаны в сети для возобновления ее работы, а также максимально быстро и легко провести поиск неисправностей.

Информация о сети должна включать:

- физические и логические схемы сети. На физической схеме должны быть показаны местоположение и соединение всех устройств сети; на логической схеме — сетевые адреса, номера сетей/подсетей;
- список сетевых протоколов с указанием информации, относящейся к ним;
- конфигурации компьютеров, серверов, коммутаторов, маршрутизаторов, межсетевых экранов.

Вполне вероятно, что в процессе поиска неисправностей в сети ее функционирование будет нарушено. Рекомендуется обязательно выделять определенный интервал времени на поиск возможных неисправностей, чтобы

10. Поиск неисправностей в сетях TCP/IP

свести к минимуму нарушение нормальной производственной деятельности. Следует документировать все внесенные изменения. Это позволит легко вернуться к прежней конфигурации, если поиск неисправности не привел к выявлению проблемы в течение времени, отведенного на техническое обслуживание.

При поиске неисправностей удобно использовать модель TCP/IP или OSI. В зависимости от типа проблемы может потребоваться выполнить одно из следующих действий:

- начать анализ работоспособности сети снизу модели TCP/IP или OSI и двигаться вверх;
- начать анализ работоспособности сети сверху модели TCP/IP или OSI и двигаться вниз.

Два первых уровня модели TCP/IP или три первых уровня модели OSI определяют функции непосредственно передачи данных. От них зависит физическая доставка сигнала по сети. Два верхних уровня модели TCP/IP или четыре верхних уровня модели OSI управляют передачей данных на уровне хост-машин.

Основные проблемы в локальной сети зачастую появляются по следующим причинам:

1. На физическом уровне:

- Отсутствие питания на оборудовании.
- Повреждение или обрыв кабеля.
- Неправильная заделка кабеля на разъем.
- Плохой контакт в месте подсоединения кабеля.
- Замыкание контактов кабеля.
- Кабель не подключен.
- Кабель подключен к неверному порту.
- Большое затухание сигнала.
- Недостаточная полоса пропускания кабеля.
- Интерференция.
- Неисправность сетевого интерфейса.

2. На канальном уровне:

- Неисправность сетевого интерфейса.
- Неисправность оборудования.
- Чрезмерная нагрузка.
- Слишком большое количество ошибок.
- Неправильная конфигурация устройства:
 - а) неверно назначенное членство в VLAN;
 - б) неверная приоритизация трафика (CoS/QoS);
 - в) неверная конфигурация протоколов STP/RSTP/MSTP, что приводит к коммутационным петлям;
 - г) неверная настройка функций безопасности, выполняющих фильтрацию по MAC-адресам.
- Проблемы с автосогласованием скоростей и режимов работы.

3. На сетевом уровне:

- Неправильное указание IP-адреса сети.
- Неверный IP-адрес сетевого интерфейса.
- Неверное указание маски подсети.
- Неверный IP-адрес сервера DNS.
- Неверный IP-адрес шлюза по умолчанию.
- Дублирование IP-адресов в сети.
- Некорректная маршрутизация.
- Неправильная конфигурация устройства:
 - а) неверная настройка протокола маршрутизации;
 - б) неверная настройка функций безопасности, выполняющих фильтрацию по IP-адресам;
 - в) неверная настройка NAT;
 - г) неверная настройка IPSec-туннеля.
- Проблема с программным обеспечением устройства.
- Исчерпание ресурсов устройства: переполнение буферов, сильная нагрузка на центральный процессор.

4. На транспортном уровне и выше:

- Некорректная настройка DHCP-сервера.
- Некорректное разрешение имен.
- Проблемы с аутентификацией.
- Неверная настройка межсетевого экрана.

10.3.1. Проверка параметров протокола IP

Предположим, что в локальной сети существует пара «источник — приемник», у которой наблюдаются проблемы с подключением. Это может быть отсутствие подключения вообще, периодическое отсутствие подключения, медленная работа сети, невозможность войти в сеть.

При подключении могут возникать следующие ошибки:

- проблема на физическом уровне;
- неверная конфигурация устройства-источника;
- проблема с получением IP-адреса по DHCP;
- дублирование IP-адресов;
- проблема с маршрутизацией внутри локальной сети;
- проблема с конфигурацией сетевых устройств.

Последовательность действий при поиске неисправности может быть следующей.

1. Проверьте, горит ли индикатор интерфейса, используемого для подключения. Если нет, то интерфейс или отключен, или неисправен. Соединение невозможно из-за отсутствия физического подключения. Исправьте подключение:

- Проверьте, что сетевой кабель подключен и с ним нет проблем.
- Проверьте, что отсутствует сбой на аппаратном уровне.
- Убедитесь, что сетевая карта компьютера не отключена.

10. Поиск неисправностей в сетях TCP/IP

Если индикатор горит, переходите к пункту 2.

2. Проверьте конфигурацию IP. На компьютере с Windows это можно сделать с помощью команды ipconfig /all (рис. 10.1). Она отображает IP-адреса, применяемые по умолчанию шлюзы и настройки DNS для всех интерфейсов. На компьютере с Linux используются следующие команды: ip addr (рис. 10.2), ip route, systemd-resolve-status.

ipconfig [/allcompartments] [/? /all /renew [адаптер] /release [адаптер] /renew6 [адаптер] /release6 [адаптер] /flushdns /displaydns /registerdns /showclassid адаптер /setclassid адаптер [идентификатор_класса]] /showclassid6 адаптер /setclassid6 адаптер [идентификатор_класса]]	
Здесь	
адаптер	Имя подключения (можно использовать знаки подстановки * and ?, см. примеры)
Параметры:	
/?	Вывод данного справочного сообщения
/all	Вывод подробных сведений о конфигурации.
/release	Освобождение адреса IPv4 для указанного адаптера.
/release6	Освобождение адреса IPv6 для указанного адаптера.
/renew	Обновление адреса IPv4 для указанного адаптера.
/renew6	Обновление адреса IPv6 для указанного адаптера.
/flushdns	Очистка кеша сопоставителя DNS.
/registerdns	Обновление всех DHCP-аренд и перерегистрация DNS-имен
/displaydns	Отображение содержимого кеша сопоставителя DNS.
/showclassid	Отображение всех допустимых для этого адаптера идентификаторов классов DHCP.
/setclassid	Изменение идентификатора класса DHCP.
/showclassid6	Отображение всех допустимых для этого адаптера идентификаторов классов DHCP IPv6.
/setclassid6	Изменение идентификатора класса DHCP IPv6.

Рис. 10.1. Параметры команды ipconfig

Отчет, выдаваемый командами, позволяет определить ошибки конфигурации IP-параметров узла. Проверьте, правильно ли указан IP-адрес, маска подсети для IPv4, IP-адрес шлюза по умолчанию, IP-адрес сервера DNS.

Если IP-адрес отсутствует, проверьте, должен ли узел-источник получать его по DHCP. Если нет, настройте адрес и другие параметры вручную. Если адрес должен быть получен через DHCP, проверьте конфигурацию DHCP-сервера, конфигурацию DHCP relay-агента, если сервер находится в другой подсети.

Если от DHCP-сервера получены неверные параметры IP, проверьте его настройки.

На управляемых коммутаторах D-Link проверить настройки IP можно с помощью команды **show ipif**.

Если проблемы в конфигурации IP отсутствуют, переходите к пункту 3.

```
ubuntu@ubuntu-pc:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether 98:4b:e1:c1:ee:95 brd ff:ff:ff:ff:ff:ff
3: wlan1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether cc:52:af:0a:9c:e1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.3/24 brd 192.168.100.255 scope global wlan1
        valid_lft forever preferred_lft forever
    inet6 fe80::ce52:afffe:fe0a:9ce1/64 scope link
        valid_lft forever preferred_lft forever
```

Рис. 10.2. Пример выполнения команды `ip addr` в Linux

3. Проверьте достижимость пункта назначения. Первым действием выполните проверку и очистку кеша ARP для IPv4. Для IPv6 выполните проверку и очистку NDP-таблицы.

На компьютере с Windows проверка и очистка кеша ARP выполняется с помощью команд **arp -a** и **arp -d**. В Linux для проверки и очистки кеша ARP используются команды **ip neigh show** и **ip neigh flush**.

На управляемом коммутаторе D-Link используются команды **show arpentry** и **clear arptable** соответственно.

Проверка и очистка NDP-таблицы на компьютере с Windows выполняется с помощью команд **netsh interface ipv6 show neighbors** и **netsh interface ipv6 delete neighbors**. В Linux применяются команды **ip -6 neigh show** и **ip -6 neigh flush**.

На управляемом коммутаторе D-Link используются команды **show ipv6 neighbor_cache** и **delete ipv6 neighbor_cache**.

Далее с помощью утилиты ping выполните проверку достижимости шлюза по умолчанию. При проверке связи со шлюзом по умолчанию определяется, можно ли достичь локальных узлов и самого шлюза, который используется для пересылки пакетов IP к узлам другой сети/подсети. Если шлюз по умолчанию фильтрует все ICMP-сообщения, выполнить этот шаг может быть невозможно.

Если шлюз не фильтрует ICMP-сообщения и результат выполнения ping сообщает об ошибке, проверьте конфигурацию шлюза по умолчанию (рис. 9.58).

10. Поиск неисправностей в сетях TCP/IP

```
C:\Users\edu1>ping 212.46.14.1

Обмен пакетами с 212.46.14.1 по с 32 байтами данных:
Ответ от 192.168.0.20: Заданный узел недоступен.

Статистика Ping для 212.46.14.1:
Пакетов: отправлено = 4, получено = 4, потеряно = 0
(0% потеря)
```

Рис. 10.3. Результат выполнения команды ping при недостижимости шлюза по умолчанию

Если конфигурация правильная, с помощью ping проверьте достижимость устройства назначения (рис. 10.4, рис. 10.5). Также выполните ping на шлюзе по умолчанию, чтобы удостовериться, может ли он установить соединение с другими устройствами сети. Отсутствие связи с другими устройствами сети говорит о проблемах с физическим подключением, неверной конфигурации коммутатора либо маршрутизатора или о дублировании IP-адресов.

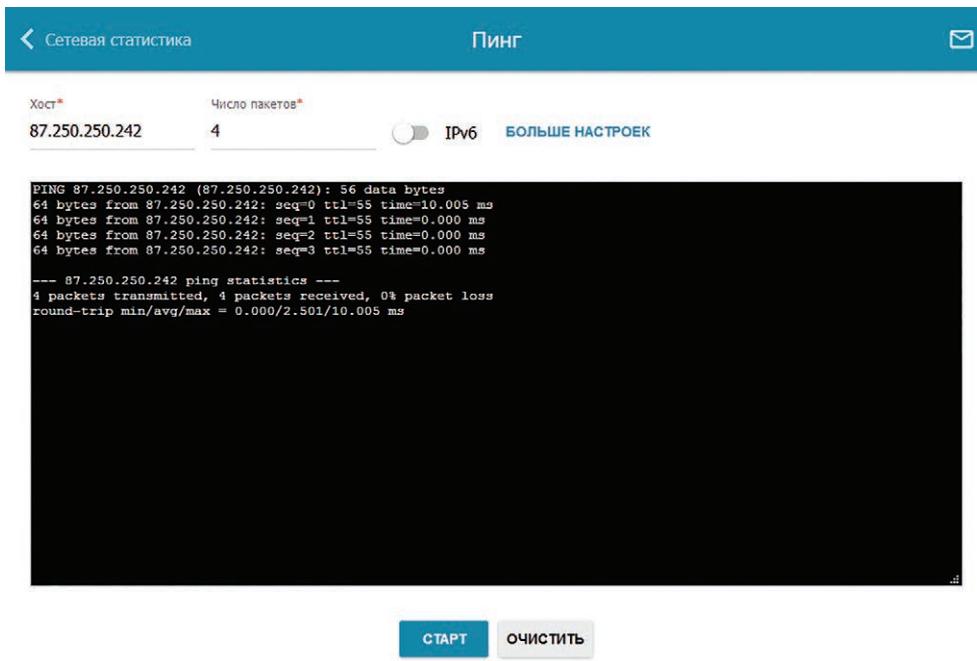


Рис. 10.4. Выполнение команды ping на маршрутизаторе D-Link

```
DGS-3120-24TC:admin#ping 192.168.0.20
Command: ping 192.168.0.20

Reply from 192.168.0.20, time=10ms
Reply from 192.168.0.20, time<10ms

Ping Statistics for 192.168.0.20
Packets: Sent =9, Received =9, Lost =0
```

Рис. 10.5. Выполнение команды ping на коммутаторе D-Link

Определить дублирование IP-адресов можно следующим образом. Отключите подозрительное устройство от сети. На любом другом устройстве выполните команду ping на этот же IP-адрес. Если результат выполнения команды успешен, значит, в сети существует еще одно устройство, использующее данный IP-адрес. Определить его MAC-адрес можно с помощью ARP-кеша или NDP-таблицы узла, на котором выполнялась команда ping.

Если после устранения дублирования IP-адресов соединение не восстановилось, необходимо выполнить проверку физического подключения.

10.3.2. Проверка физического соединения

Большинство современных локальных сетей построено на коммутаторах. Основой коммутируемой среды является порт подключения, поэтому все ответы на вопросы «Что произошло?», «Как до этого работало?» и другие должны относиться непосредственно к порту. Описание в этом разделе будет приведено для устройств D-Link.

Некоторые порты имеют особую важность, так как подключаются к другим коммутаторам, серверам и маршрутизаторам. Через них передается большое количество данных.

На работу портов влияют:

- аппаратные факторы;
- проблемы с конфигурацией;
- чрезмерный трафик.

С чего следует начать, когда выясняется, что в коммутируемой сети произошла проблема?

10. Поиск неисправностей в сетях TCP/IP

Сначала проверьте, включено ли питание коммутатора или маршрутизатора, к которому подключено устройство, испытывающее проблемы с соединением.

Если питание включено, проверьте работу портов. Для того чтобы порт коммутатора или маршрутизатора работал, он должен быть соединен исправным кабелем соответствующего типа и требуемой длины с другим устройством.

Проверьте на коммутаторе индикаторы Link/Act портов (рис. 10.6). Когда подключен исправный кабель и соединение установлено на максимальной для этого порта скорости, индикатор будет гореть зеленым светом. При передаче данных индикатор порта будет мигать.



Рис. 10.6. Индикация портов коммутатора D-Link

При подключении кабеля к порту маршрутизатора индикатор LAN или Интернета соответствующего порта будет гореть зеленым или голубым светом в зависимости от модели. При передаче данных индикатор порта будет мигать.

Если кабель не подключен, сетевая розетка неисправна или порт по каким-то причинам не работает, индикатор гореть не будет. Убедитесь, что кабель исправлен и хорошо вставлен в разъем порта и сетевой розетки, если подключение выполняется не напрямую. Может оказаться так, что кабель вставлен не в тот порт. Подключите кабель в правильный порт и проверьте наличие соединения.

Если кабель исправен, вставлен правильно, но соединение не установлено, проверьте настройки соответствующего порта. Порт мог быть

Port	State/ MDIX	Settings Speed/Duplex/FlowCtrl	Connection Speed/Duplex/FlowCtrl	Address Learning	AutoSpeed Downgrade
1	Enabled Auto	Auto/Disabled	1000M/Full/None	Enabled	Disabled
2	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
3	Enabled Auto	100M/Full/Disabled	100M/Full/None	Enabled	Disabled
4	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
5	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
6	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
7	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
8	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled
9	Enabled Auto	Auto/Disabled	Link Down	Enabled	Disabled

Рис. 10.7. Результат выполнения команды show ports

по какой-то причине отключен. На коммутаторе D-Link с интерфейсом командной строки можно посмотреть настройки порта с помощью команды **show ports** (рис. 10.7).

Порт	Статус	Автосогласование	Скорость	Контроль потока
LAN1	Подключено	Включено	1000M-Full	802.3x(tx+rx)
LAN2	Отключено	Отключено	-	-
LAN3	Подключено	Включено	100M-Half	Back-pressure
LAN4	Отключено	Отключено	-	-
WAN	Отключено	Отключено	-	-

Рис. 10.8. Вкладка Настройки портов

На маршрутизаторе D-Link информацию о состоянии портов можно посмотреть во вкладках *Настройки портов* (рис. 10.8) Web-интерфейса управления и *Информация о системе* (рис. 10.9).

10. Поиск неисправностей в сетях TCP/IP

The screenshot shows the 'Information about the system' tab with the following sections:

- Информация о системе**:
 - Модель: DIR-825ACG1
 - Аппаратная ревизия: G1A
 - Версия ПО: 3.0.3
 - Время сборки: Wed Oct 11 12:18:52 MSK 2017
 - Производитель: D-Link Russia
 - Тех. поддержка: support@dlink.ru
 - Описание: Root filesystem image for DIR-825ACG1
 - Время работы: 0 дней 00:48:25
 - Режим работы: Router
 - Включить светодиодные индикаторы:
- WAN по IPv6**:
 - Тип соединения: Динамический IPv6
 - Статус: Кабель отключен
- Локальная сеть**:
 - LAN IPv4: 192.168.0.1
 - LAN IPv6: 3004: ...:fe9e:74f0/64
 - Беспроводные подключения: -
 - Проводные подключения: 2
- Порты LAN**:
 - LAN1: 1000M-Full
 - LAN2: Отключено
 - LAN3: 100M-Full
 - LAN4: Отключено
- Wi-Fi 2.4 ГГц**:
 - Статус: Включено
 - Вещание: Включено
 - Количество дополнительных сетей: 0
 - Имя сети (SSID): DIR-825-74ef
 - Безопасность: WPA2-PSK
- Wi-Fi 5 ГГц**:
 - Статус: Включено
 - Вещание: Включено
 - Количество дополнительных сетей: 0
 - Имя сети (SSID): DIR-825-5G-74ef
 - Безопасность: WPA2-PSK
- USB-устройства**:

Нет подключенных устройств
- Яндекс.DNS**:
 - Вкл/выкл: Включить
 - Безопасный: 6 устройств
 - Детский: 0 устройств
 - Без защиты: 0 устройств

Рис. 10.9. Вкладка Информация о системе

Индикатор порта коммутатора будет гореть оранжевым светом, если соединение установлено на скорости меньше максимальной. Например, если на порте Gigabit Ethernet установлено соединение на скорости 10/100 Мбит/с.

В том случае, если на другом конце канала связи находится устройство, поддерживающее соединение на аналогичной максимальной скорости, проверьте, исправен ли кабель. Если кабель неисправен, замените его.

Если кабель исправен, но соединение на максимальной скорости не устанавливается, проверьте конфигурацию портов подключения. Может оказаться, что возникла проблема с автосогласованием скорости и режима работы или на одном из портов настроено ограничение скорости соединения.

Несоответствие настроек режима работы (дуплекс/половинный) не вызовет сбоя подключения. Если настройки режима работы не совпадают, то будут появляться ошибки, сеть будет работать медленнее, что нетрудно заметить визуально, но трафик все-таки будет передаваться от одного устройства другому.

Если на порте наблюдается большое количество ошибок FCS (Fragment), коллизий — это говорит о несогласованности режимов работы. Другие типы ошибок вызывают проблемы с кабелем или сетевым интерфейсом. Если появляется большое количество отложенных кадров (Excessive Deferal), то в данном сегменте, скорее всего, избыточное количество трафика и буферы порта заполнены.

Port Number : 1			
	RX Frames		TX Frames
CRC Error	0	Excessive Deferral	0
Underrun	0	CRC Error	0
Oversize	0	Late Collision	0
Fragment	0	Excessive Collision	0
Jabber	0	Single Collision	0
Drop Pkts	53	Collision	0
Symbol Error	0		

Рис. 10.10. Результат выполнения команды show error ports

Посмотреть статистику ошибок на порте коммутатора можно с помощью команды **show error ports** (рис. 10.10).

На маршрутизаторе статистика доступна во вкладке *Подробная информация о порте* (рис. 10.11).

10. Поиск неисправностей в сетях TCP/IP

◀ Статистика портов Подробная информация о порте ⓐ

LAN1

Основные счетчики	
Отправлено трафика (байт)	2423041
Получено трафика (байт)	1582388
Отправлено пакетов	10458
Получено пакетов	13070
Отправлено unicast пакетов	9680
Получено unicast пакетов	11196
Отправлено multicast пакетов	546
Получено multicast пакетов	1396
Отправлено broadcast пакетов	272
Получено broadcast пакетов	519
Количество пакетов, отброшенных при отправке	0
Количество пакетов, отброшенных при получении	122
Количество ошибок при отправке	0
Количество ошибок при получении	0
Получено пакетов неизвестного протокола	0

Дополнительные счетчики для принятых пакетов	
Пакеты меньше 64 байт	0
Пакеты 64 байта	8001
Пакеты от 65 до 127 байт	10408
Пакеты от 128 до 255 байт	1019
Пакеты от 256 до 511 байт	1770
Пакеты от 512 до 1023 байт	854
Пакеты от 1024 до 1518 байт	716
Пакеты больше 1518 байт	0
Fragments ① Пакеты меньше 64 байт с некорректным значением FCS	0
Пакеты с CRC ошибками ① Пакеты от 64 до 1518 байт с некорректным значением FCS	0
Jabbers ① Пакеты больше 1518 байт с некорректным значением FCS	0
Drop events ① Общее количество событий, при которых пакеты были отброшены из-за недостатка ресурсов	0

Контроль потока	
Отправлено фреймов PAUSE	0
Получено фреймов PAUSE	0

Рис. 10.11. Вкладка Подробная информация о порте

10.3.3. Проверка канального уровня

Если после проверки физического соединения и устранения найденных неполадок связь не восстановлена, возможно, проблема заключается в невозможности кадров проходить через коммутатор или маршрутизатор. Об этом говорит большое количество отброшенных портом кадров (Drop Pkts или Drop events в статистике ошибок).

Одной из проблем может являться неправильная настройка при использовании виртуальных локальных сетей (VLAN). Порт, к которому подключен источник, и порт, к которому подключен пункт назначения, могут находиться в разных VLAN. Для передачи трафика между разными VLAN требуется маршрутизация. Однако если в сети не предусмотрена маршрутизация и VLAN используются для изоляции трафика разных групп пользователей, необходимо проверить настройки портов подключения.

Зачастую интерфейсные карты серверов поддерживают 802.1Q VLAN. Возможной причиной, по которой пользователь не может подключиться к серверу, является некорректная настройка магистрального порта коммутатора или интерфейса сервера. Удостоверьтесь, что магистральные порты коммутаторов и интерфейсы серверов являются маркированными членами тех VLAN, чей трафик они должны принимать и передавать.

Также может оказаться, что какое-то устройство подключено не в тот порт (с другими настройками VLAN) или настройки порта неверны (он является членом не той VLAN, тип порта (маркированный/немаркированный) указан неверно или настройки VLAN отсутствуют).

С помощью утилиты **ping** проверьте достижимость различных устройств, находящихся в одной VLAN. Если какие-либо устройства недостижимы, проверьте настройки VLAN на соответствующих портах.

На коммутаторе можно D-Link проверить настройки VLAN с помощью команды **show vlan** (рис. 10.12).

Также полезно проверить таблицу коммутации с помощью команды **show fdb**. Таблица коммутации является структурой, на основе которой коммутатор принимает решение о перенаправлении трафика. В ней перечислены VLAN, динамически изученные MAC-адреса и порты. С ее помощью можно выяснить, является ли порт членом нужной VLAN, и проверить, правильно ли порт изучает MAC-адреса.

Проблемы с изучением MAC-адресов могут возникнуть в случае некорректной работы протоколов семейства STP. Проверить их настройки на коммутаторе D-Link можно с помощью команды **show stp**.

На порту коммутатора могут быть активированы функции контроля подключения (списки управления доступом, Port Security, IMPB и др.). В этом случае также могут возникать проблемы с прохождением трафика через коммутатор. Проверьте настройки соответствующих функций, а также Log-файл коммутатора с помощью команды **show log** (рис. 10.13), если запись событий в него была активирована.

10. Поиск неисправностей в сетях TCP/IP

```
VID : 2          VLAN Name : v2
VLAN Type : Static   Advertisement : Disabled
Member Ports : 1-2
Static Ports : 1-2
Current Tagged Ports :
Current Untagged Ports: 1-2
Static Tagged Ports :
Static Untagged Ports : 1-2
Forbidden Ports :

VID : 3          VLAN Name : v3
VLAN Type : Static   Advertisement : Disabled
Member Ports : 3-4
Static Ports : 3-4
Current Tagged Ports :
Current Untagged Ports: 3-4
Static Tagged Ports :
Static Untagged Ports : 3-4
Forbidden Ports :
```

Рис. 10.12. Результат выполнения команды show vlan

Index	Date	Time	Level	Log Text
8	2018-05-11	10:09:28	INFO(6)	Port 3 link up, 100Mbps HALF duplex
7	2018-05-11	10:07:24	INFO(6)	Port 3 link down
6	2018-05-11	09:58:06	INFO(6)	Port 3 link up, 100Mbps HALF duplex
5	2018-05-11	09:58:03	INFO(6)	Port 3 link down
4	2018-05-11	09:47:20	INFO(6)	Port 1 link up, 1000Mbps FULL duplex
3	2018-05-11	09:47:08	INFO(6)	Port 1 link down
2	2018-05-11	09:46:10	INFO(6)	Port 3 link up, 100Mbps FULL duplex
1	2018-05-11	09:46:07	INFO(6)	Port 3 link down

Рис. 10.13. Результат выполнения команды show log

10.3.4. Проверка сетевого уровня

Когда пакет маршрутизируется через большую сеть, различные проблемы, приводящие к потере соединения, могут возникнуть на любом узле между источником и приемником.

С помощью утилит трассировки **tracert** (Windows) (рис. 10.14) или **traceroute** (Linux) можно проследить путь передачи пакетов IP к узлу назначения. Принцип работы этих утилит следующий. Для исследования пути передачи пакетов IPv4 источник отправляет целевому узлу серию запросов ICMP Echo Request, увеличивая значение поля TTL на 1 для каждого последующего пакета. Первоначально пакет отправляется со значением поля TTL,

равным 1. Каждый маршрутизатор на пути обязан уменьшить значение TTL пакета на 1 перед его дальнейшей пересылкой. Первый на пути маршрутизатор вычтет 1 из значения TTL, и оно станет равным нулю (0). В результате пакет будет отброшен, и маршрутизатор отправит источнику ICMP-сообщение *Time Exceeded*.

Команда трассировки узнает IP-адрес первого на пути маршрутизатора из полученного ICMP-сообщения. Этот адрес, а также время между отправкой пакета и получением ответа выводятся на монитор компьютера. Значение поля TTL следующего отправляемого источником пакета увеличивается на 1. Процесс повторяется до тех пор, пока узел назначения не ответит или не будет достигнуто максимальное число прыжков.

Процесс трассировки завершается, когда от узла-назначения получен ответ ICMP Echo Reply.

При исследовании пути передачи пакетов IPv6 используется такой же принцип. Отличие в том, что вместо поля TTL используется поле Hop Limit.

```
C:\>tracert google.ru

Трассировка маршрута к google.ru [172.217.17.35]
с максимальным числом прыжков 30:

 1   1 ms    1 ms    1 ms  192.168.100.1
 2   3 ms    2 ms    2 ms  10.159.0.1
 3  16 ms    4 ms    3 ms  10.109.11.6
 4   5 ms    4 ms    4 ms  10.109.11.18
 5  13 ms    5 ms    7 ms  mag9-cr01-be12.51.msk.stream-internet.net [212.188.1.5]
 6  18 ms    5 ms    4 ms  72.14.223.74
 7   6 ms    5 ms    7 ms  108.170.250.81
 8  26 ms   28 ms   26 ms  72.14.236.248
 9  44 ms   45 ms   45 ms  108.170.235.71
10  46 ms   46 ms   46 ms  216.239.47.198
11  47 ms   46 ms   46 ms  209.85.254.10
12  46 ms   47 ms   48 ms  209.85.255.215
13  48 ms   48 ms   48 ms  108.170.241.225
14  47 ms   47 ms   47 ms  108.170.236.137
15  46 ms   46 ms   46 ms  ams16s29-in-f35.1e100.net [172.217.17.35]

Трассировка завершена.
```

Рис. 10.14. Результат выполнения команды tracert

Если на пути между источником и приемником промежуточные узлы фильтруют ICMP-пакеты, трассировку выполнить невозможно. На экран будет выводиться сообщение «Превышен интервал ожидания для запроса».

Одной из причин невозможности достижения локального или удаленного узла назначения может быть неправильность или отсутствие маршрутов в таблице маршрутизации. Проверьте и скорректируйте, если необходимо, таблицы маршрутизации устройств локальной сети. Если проблемы находятся за ее пределами, обратитесь к поставщику услуг.

10. Поиск неисправностей в сетях TCP/IP

Для просмотра таблицы маршрутизации на рабочей станции под управлением ОС Windows служат команды **route print** или **netstat -r**. В Linux используются команды **netstat -r** или **ip route**. Убедитесь в наличии маршрута по умолчанию в таблице маршрутизации, а также в присутствии маршрута к нужному узлу или сети. При использовании IPsec иногда необходимо создать статический маршрут к какому-то узлу в удаленной сети.

Для добавления отсутствующего маршрута к таблице маршрутизации на рабочей станции используйте команду **route add** (Windows) или **ip route add** (Linux).

Еще одной причиной отсутствия соединения в большой сети могут быть петли маршрутизации, вызванные неправильной настройкой протоколов маршрутизации, неверноенным статическим маршрутом или отсутствием физического соединения.

Петлю маршрутизации можно определить с помощью команды трассировки. На рис. 10.15 показан результат выполнения команды **tracert** в сети с петлей маршрутизации.

```
C:\Users\edu1>tracert 192.168.5.2

Трассировка маршрута к 192.168.5.2 с максимальным числом прыжков 30

 1  1 ms    1 ms    1 ms  192.168.2.1
 2  *        *        *        Превышен интервал ожидания для запроса.
 3  1 ms    1 ms    1 ms  192.168.3.1
 4  *        *        *        Превышен интервал ожидания для запроса.
 5  1 ms    1 ms    1 ms  192.168.3.1
 6  *        *        *        Превышен интервал ожидания для запроса.
 7  1 ms    1 ms    1 ms  192.168.3.1
 8  *        *        *        Превышен интервал ожидания для запроса.
 9  1 ms    1 ms    1 ms  192.168.3.1
10  *        *        *        Превышен интервал ожидания для запроса.
11  1 ms    1 ms    1 ms  192.168.3.1
12  *        *        *        Превышен интервал ожидания для запроса.
13  1 ms    1 ms    1 ms  192.168.3.1
```

Рис. 10.15. Петля маршрутизации

При проблемах с маршрутизацией в локальной сети проверьте настройки всех маршрутизирующих устройств.

Начните поиск проблемы с изучения таблиц маршрутизации. На коммутаторе это можно сделать с помощью команды **show iproute**.

Обратите внимание, имеются ли в таблице статические маршруты. Они создаются администратором вручную и могут содержать ошибки. Например, неверно указан IP-адрес или маска сети узла назначения, IP-адрес следующего на пути маршрутизатора.

Проверьте настройки протоколов маршрутизации, используемых в сети. При использовании протоколов RIP возможны следующие причины неполадок:

- Использование разных версий протокола RIP на маршрутизирующих устройствах;
 - Протокол глобально не активирован на устройстве;
 - Отправитель и получатель обновлений RIP находятся в разных подсетях. Например, на одном из интерфейсов неправильно указана маска подсети;
 - Использование аутентификации в RIPv2. Если ее параметры не совпадают, обновления будут отбрасываться;
 - Неверная конфигурация сети. В сети имеются узлы, расстояние между которыми превышает 15 переходов;
 - На маршрутизирующих устройствах отличаются настройки таймеров RIP.

При использовании в сети протокола OSPF возможны следующие причины неполадок:

- Протокол глобально не активирован на устройстве;
- Интервалы Hello и Dead у маршрутизаторов, подключенных к одной сети, не совпадают;
- На разных концах канала связи не совпадает Area ID;
- Не совпадают настройки аутентификации, если она используется;
- В сети дублируется Router ID. Он должен быть уникальным;
- У маршрутизаторов, которые должны установить соседство, не совпадает тип сети OSPF;
- Маршрутизаторы, которые должны установить соседство, находятся в разных подсетях.

Также на функционирование маршрутизации может влиять наличие фильтров, настроенных на устройствах сети. Проверьте фильтры или списки управления доступом на коммутаторах, маршрутизаторах и межсетевых экранах.

Еще одной возможной проблемой при прохождении IP-пакетов является некорректная работа NAT. Проверьте его настройки на маршрутизаторах и межсетевых экранах, установленных в сети.

10.3.5. Проверка протоколов верхних уровней

Прежде чем переходить к поиску неисправностей на верхних уровнях, необходимо убедиться, что соединение на сетевом уровне работает исправно.

Фильтрация пакетов по исходному узлу, промежуточным маршрутизаторам и узлу назначения может препятствовать созданию соединений TCP.

В ряде случаев фильтрация пакетов настраивается с целью разрешить определенные типы трафика и запретить все остальные либо запретить определенные типы трафика и разрешить все остальные.

10. Поиск неисправностей в сетях TCP/IP

Можно использовать команду **telnet IPv4-адрес TCP-порт** или **telnet IPv6-адрес TCP-порт**, чтобы убедиться в возможности установления TCP-соединения с узлом назначения с помощью известного номера порта. Например, чтобы проверить, принимает ли TCP-соединения служба Web-сервера на устройстве с IPv4-адресом 192.168.100.1, используйте команду **telnet 192.168.100.1 80**.

Если утилите telnet удается установить TCP-соединение, окно командной строки очищается и в зависимости от протокола отображается некоторый текст. В этом окне можно ввести команды для службы, к которой установлено подключение. Если утилите telnet не удается создать TCP-подключение, отображается сообщение, показанное на рис. 10.16.

```
C:\>telnet 192.168.100.12 80
Подключение к 192.168.100.12...Не удалось открыть подключение к этому узпу, на порт 80: Сбой подключения
```

Рис. 10.16. Пример работы команды telnet

Удобным средством исследования сети и сканирования портов является бесплатная утилита с открытым кодом Nmap. В настоящее время доступны ее версии для Linux, Windows и Mac OS X. Помимо командной строки набор Nmap включает графический интерфейс (Zenmap), гибкий инструмент передачи, перенаправления и отладки данных (Ncat), утилиту для сравнения результатов сканирования (Ndiff) и инструмент генерации пакетов и анализа ответов (Nping).

Nmap использует сырые сокеты для определения хостов, доступных в сети, предлагаемых ими служб, используемых операционных систем, типов пакетных фильтров/межсетевых экранов и еще множества других характеристик.

Команда имеет следующий синтаксис:

```
nmap [ <Тип сканирования> ... ] [ <Опции> ]
          { <цель сканирования> }
```

Выходные данные Nmap — это список просканированных целей с дополнительной информацией по каждой в зависимости от заданных опций. Одним из результатов сканирования является таблица портов. Она содержит номер порта, его состояние, протокол и имя службы.

Nmap распознает следующие состояния портов:

- Open: порт открыт. Приложение принимает запросы на установку TCP-соединения или UDP-дейтаграммы на этот порт. Обнаружение этого состояния обычно является основной целью сканирования. Открытые порты позволяют определить службы доступные в сети;
- Closed: порт закрыт. Закрытый порт доступен (он принимает и отвечает на запросы Nmap), но не используется каким-либо приложением;
- Filtered: порт фильтруется. Межсетевой экран или пакетный фильтр блокирует порт, и Nmap не может установить, открыт он или закрыт;

- Unfiltered: порт не фильтруется. Это состояние соответствует случаю, когда порты отвечают на запросы, но Nmap не может определить, открыты они или закрыты.

Nmap выдает комбинации состояний **open|filtered** и **closed|filtered**, если не может определить, какое из них описывает порт. По запросу в таблице портов также может отображаться информация о версии программного обеспечения.

При сканировании протокола IP (-sO) утилита Nmap не предоставляет информацию об открытых портах. Выдается только информация о протоколе IP и поддерживающих его протоколах.

В дополнение к таблице важных портов Nmap может предоставлять следующую информацию о целях: преобразованные имена DNS, предположение об используемой операционной системе, типы устройств и MAC-адреса.

Типичное сканирование с использованием командной строки Nmap и графического интерфейса Zenmap показано на рис. 10.17 и рис. 10.18. В примере используются аргументы **-A** (определение версии ОС, сканирование с использованием скриптов и трассировка) и **-T4** (ускоренное выполнение).

```
$ nmap 192.168.0.10

Starting Nmap 7.60 ( https://nmap.org ) at 2018-07-25 17:06 MSK
Nmap scan report for 192.168.0.10
Host is up (0.00019s latency).
Not shown: 992 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
49/tcp    open  tacacs
80/tcp    open  http
111/tcp   open  rpcbind
873/tcp   open  rsync
1723/tcp  open  pptp
9418/tcp  open  git

Nmap done: 1 IP address (1 host up) scanned in 13.07 seconds
```

Рис. 10.17. Пример сканирования с помощью Nmap

10. Поиск неисправностей в сетях TCP/IP

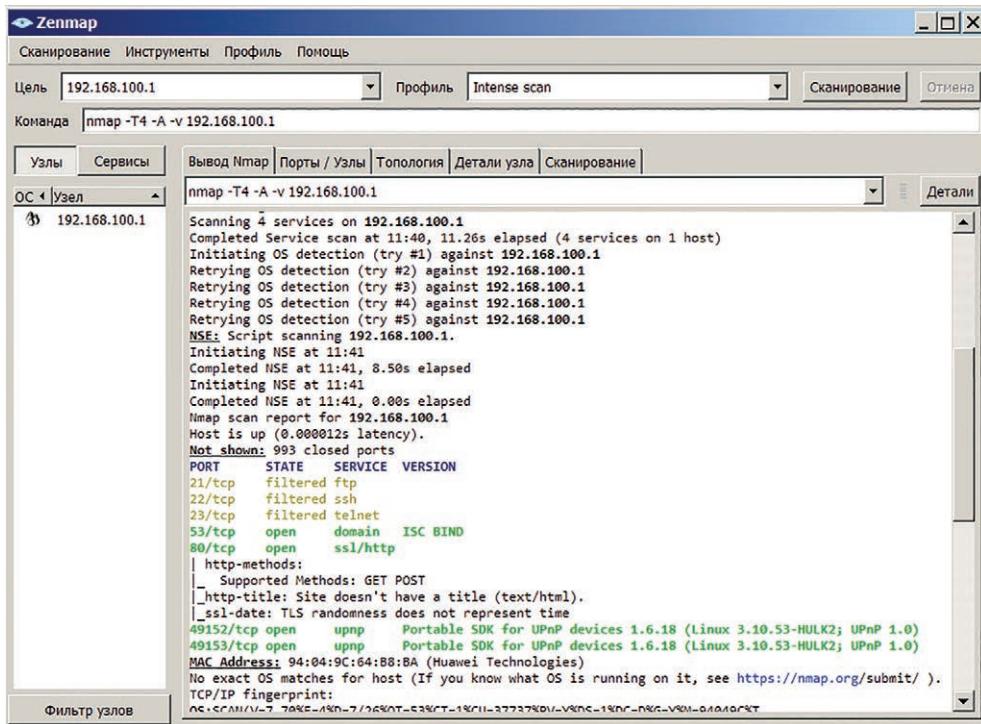


Рис. 10.18. Пример сканирования с помощью Zenmap

Утилита Nmap может сканировать как один узел, так и целую сеть. Имеются опции для определения портов для сканирования и его порядка: произвольного или последовательного.

Помимо фильтрации трафика, проблемы на верхних уровнях могут быть связаны с невозможностью входа пользователя в сеть. Одной из причин этого является неверная настройка протокола аутентификации.

Неправильная работа службы DNS также приводит к проблемам соединения на верхних уровнях. Если достижимость с использованием IP-адресов возможна, а с использованием имен узлов — нет, то может иметь место проблема с разрешением имен узлов, которая обычно возникает из-за неправильной настройки клиента DNS или вследствие проблем с регистрацией DNS.

Для того чтобы протестировать разрешение имен DNS, воспользуйтесь утилитой ping, с помощью которой можно проверить связь с объектом назначения по имени его узла или полному доменному имени. Утилита ping отображает доменное имя и его разрешенный IPv4-адрес. Если на узле, на котором запускается ping, используется как протокол IPv4, так и IPv6, запрос DNS возвращает и IPv4-адреса, и IPv6-адреса. Если связь с пунктом

назначения по его доменному имени не устанавливается, ping сообщает о том, что узел не удалось обнаружить.

Имейте в виду, что если в сети присутствуют устройства, которые фильтруют все ICMP-сообщения, выполнить этот шаг может быть невозможно.

В Windows и Linux для проверки и устранения неполадок DNS можно воспользоваться утилитой **nslookup** (рис. 10.19). Она работает в режиме командной строки и позволяет выполнять быстрое преобразование из доменного имени в IP-адрес и наоборот.

Существуют два режима использования утилиты nslookup: интерактивный и неинтерактивный. Чтобы запустить ее в интерактивном режиме, следует просто ввести nslookup в командной строке без параметров. Если данные, введенные в командной строке, не являются правильной командой утилиты nslookup, то они рассматриваются как имя хоста и происходит попытка распознать это имя. Для завершения работы утилиты в интерактивном режиме следует либо нажать комбинацию клавиш <CTRL + C>, или ввести команду exit либо quit в командной строке.

```
C:\>nslookup
Default Server: UnKnown
Address: 192.168.16.90

> dlink.ru
Server: UnKnown
Address: 192.168.16.90

Non-authoritative answer:
Name: dlink.ru
Address: 178.170.168.19

> google.ru
Server: UnKnown
Address: 192.168.16.90

Non-authoritative answer:
Name: google.ru
Addresses: 2a00:1450:4010:c09::5e
           173.194.220.94

> exit
```

Рис. 10.19. Утилита nslookup

Утилита **netstat** (рис. 10.20) отображает статистику протоколов и текущих сетевых подключений TCP/IP. Именно к ней чаще всего обращаются системные администраторы, чтобы быстро отыскать причину неисправности в сети TCP/IP. Выдаваемая информация зависит от операционной системы, но обычно выводятся такие данные: список соединений, статистика сетевых интерфейсов, информация о буферах данных, содержание таблицы маршрутизации, статистика работы протоколов.

Тип выводимой информации можно выбирать с помощью опций командной строки.

10. Поиск неисправностей в сетях TCP/IP

C:\>netstat ?	
Отображение статистики протокола и текущих сетевых подключений TCP/IP.	
NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p протокол] [-r] [-s] [-t] [интервал]	
-a	Отображение всех подключений и ожидающих портов.
-b	Отображение исполняемого файла, участвующего в создании каждого подключения, или ожидающего порта. Иногда известные исполняемые файлы содержат множественные независимые компоненты. Тогда отображается последовательность компонентов, участвующих в создании подключения, либо ожидающий порт. В этом случае имя исполняемого файла находится снизу в скобках [], сверху - компонент, который им вызывается, и так до тех пор, пока не достигается TCP/IP. Заметьте, что такой подход может занять много времени и требует достаточных разрешений.
-e	Отображение статистики Ethernet. Может применяться вместе с параметром -s.
-f	Отображение полного имени домена (FQDN) для внешних адресов.
-n	Отображение адресов и номеров портов в числовом формате.
-o	Отображение кода (ID) процесса каждого подключения.
-p протокол	Отображение подключений для протокола, задаваемых этим параметром. Допустимые значения: TCP, UDP, TCPv6 или UDPv6. Используется вместе с параметром -s для отображения статистики по протоколам. Допустимые значения: IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP или UDPv6.
-r	Отображение содержимого таблицы маршрутов.
-s	Отображение статистики протокола. По умолчанию статистика отображается для протоколов IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP и UDPv6. Параметр -p позволяет указать подмножество выводимых данных.
-t	Отображение текущего подключения в состоянии "offload".
интервал	Повторный вывод статистических данных через указанный

Рис. 10.20. Параметры утилиты netstat

В Linux для просмотра текущих соединений, открытых и прослушиваемых портов используется утилита ss (рис. 10.21).

Синтаксис команды следующий:

```
$ ss опции [фильтр_состояния] [фильтр_адреса]
```

Приведем некоторые из опций, которые можно использовать для анализа сетевых подключений:

- n, -- numeric — не разрешать имена DNS;
- r, -- resolve — разрешать имена DNS;
- a, -- all — показать все сокеты (открытые соединения);
- l, -- listening — показать только прослушиваемые сокеты;
- e, -- extended — показать подробную информацию о сокете;
- p, -- processes — показать процессы, использующие сокет;
- i, -- internal — показать внутреннюю информацию протокола TCP;
- s, -- summary — показать статистику использования сокета;
- 4, -- IPv4 — показать только информацию о сокетах протокола IPv4;

- 6, -- IPv6 — показать только информацию о сокетах протокола IPv6;
- t, -- TCP — показать только информацию о сокетах протокола TCP;
- u, -- UDP — показать только информацию о сокетах протокола UDP;
- d, -- DHCP — показать только информацию о сокетах протокола DHCP.

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
u_seq	ESTAB	0	0	* 33476	* 33477
u_str	ESTAB	0	0	* 33231	* 34025
u_str	ESTAB	0	0	@/tmp/.X11-unix/X0 33156	* 33931
u_str	ESTAB	0	0	* 266631	* 264729
u_str	ESTAB	0	0	* 40713	* 41778
u_str	ESTAB	0	0	* 32999	* 33815
u_str	ESTAB	0	0	/run/user/1000/bus 77632	* 74572

Рис. 10.21. Результат выполнения команды ss без опций

ГЛОССАРИЙ

А

ACL (англ. Access Control List). Списки управления доступом. Списки управления доступом являются средством фильтрации потоков данных на аппаратном уровне. Используя ACL можно ограничить типы приложений, разрешенных для использования в сети, контролировать доступ пользователей к сети и определять устройства, к которым они могут подключаться. Также ACL могут использоваться для определения политики QoS путем классификации трафика и переопределения его приоритета.

ADSL (англ. Asymmetric Digital Subscriber Loop). Асимметричный цифровой абонентский контур. Технология семейства xDSL, обеспечивающая передачу данных через тот же локальный телефонный контур, по которому предоставляются услуги обычной аналоговой телефонии.

AES (англ. Advanced Encryption Standard). Симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит). Инициатива в разработке алгоритма AES принадлежит Национальному институту стандартов и технологий (NIST) США. В результате длительного процесса оценки предложенных алгоритмов в качестве алгоритма AES был выбран алгоритм Rijndael. Алгоритм AES определен в FIPS PUB 197-2001. Он был адаптирован под требования многих протоколов, включая протокол CCMP (CTR with CBC-MAC Protocol) для сетей 802.11.

AH (англ. Authentication Header). Один из протоколов IPsec, который обеспечивает целостность протоколов, расположенных выше в стеке протоколов, и целостность отдельных полей IP-заголовка, которые не изменяются при пересылке от отправителя к получателю, дополнительно может обеспечиваться анти-replay сервис, т. е. целостность некоторой последовательности дейтаграмм. В IPsecv3 реализация данного протокола не является обязательной.

ARP (англ. Address Resolution Protocol). Протокол разрешения адресов. Протокол, используемый для динамического преобразования IP-адресов в физические (аппаратные) MAC-адреса устройств локальной сети TCP/IP. В общем случае ARP требует передачи широковещательного сообщения всем узлам, на которое отвечает узел с соответствующим запросу IP-адресом.

Asymmetric chipper. Асимметричное шифрование. Криптосистема, в которой шифрование и расшифровывание выполняются с использованием двух разных ключей, один из которых называется открытым, а другой — закрытым. Важным свойством этой системы является то, что, зная открытый ключ, вычислительно невозможно определить закрытый ключ. Асимметричные криптосистемы могут использоваться для шифрования, подписи, а также для обмена ключа.

Глоссарий

Authentication. Аутентификация. Сервис безопасности, который обеспечивает подтверждение того, что информация получена от законного источника и получатель является требуемым.

Authorisation. Авторизация. Предоставление прав доступа. Права и разрешения, предоставленные индивидууму (или процессу), которые обеспечивают возможность доступа к ресурсу. После того как пользователь аутентифицирован, авторизация определяет, какие права доступны пользователю.

Auto-negotiation. Автосогласование. Функция, обеспечивающая механизм автоматической настройки портов многоскоростных устройств. Устройства, поддерживающие функцию автосогласования, могут определять режимы работы партнеров по соединению, оповещать их о своих режимах работы и выбирать наилучший режим для совместного функционирования.

AS (англ. Autonomous system). Автономная система. Представляет собой группу маршрутизаторов и IP-сетей, контролируемых одной организацией или находящихся под единым административным управлением, которые используют единую согласованную политику для внутренней маршрутизации.

В

Backbone. Магистраль. Часть сети, по которой передается основной трафик и которая чаще всего является источником и приемником трафика других сетей.

Broadcast. Широковещание. Система доставки пакетов, при которой копия каждого пакета передается всем узлам, подключенным к сети.

Broadcast storm. Широковещательный штурм. Множество одновременных широковещательных рассылок в сети, которые, как правило, поглощают доступную полосу пропускания сети и могут вызвать отказ сети.

С

CA (англ. Certification Authority). Сертификационный центр. Уполномоченный орган, который выпускает сертификаты.

CHAP (англ. Challenge Handshake Authentication Protocol). Протокол используется для периодической проверки аутентификации противоположной стороны с помощью трехстороннего рукопожатия (3-way handshake). Выполняется после установления канала и может быть повторно использован в любое время после того, как канал установлен. Регламентируется RFC 1334, RFC 1994.

Collision. Коллизия. Наложение или столкновение сигналов, которое возникает во время одновременной передачи данных двумя или более узлами и приводит к повреждению данных.

CoS (англ. Class of Service). Класс обслуживания. Способ классификации и приоритизации пакетов на основе типа приложения или других методов

классификации (802.1p, ToS, DiffServ) для обеспечения качества обслуживания в сети.

CPE (англ. Customer Premises Equipment). Абонентское устройство. Пользовательское (оконечное) оборудование, подключаемое к сети связи.

D

Data confidentiality. Конфиденциальность данных. Сервис безопасности, который обеспечивает недоступность информации неавторизованным способом.

DES (англ. Data Encryption Standard). Алгоритм симметричного шифрования, длина блока равна 64 битам, длина ключа равна 56 битам.

DH (англ. Diffie-Hellman). Алгоритм Диффи — Хеллмана. Алгоритм асимметричного шифрования, основанный на использовании неразрешимости задачи дискретного логарифмирования. Используется для обмена ключа.

DHCP (англ. Dynamic Host Configuration Protocol). Протокол динамической конфигурации узла. Сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Является расширением протокола BOOTP. Регламентируется RFC 2131 и др.

Digital signature. Цифровая подпись. Блок данных, являющийся результатом функции подписывания, на вход которой подаются закрытый ключ подписывающей стороны (ключ подписывания), сообщение (или хеш-функция от сообщения) и параметры домена. Данный блок данных позволяет получателю проверить целостность данных и источник данных.

DNS (англ. Domain Name System). Система доменных имен. Компьютерная распределенная система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене.

DoS (англ. Denial-of-service). Атака типа «отказ в обслуживании». Отказ в обслуживании нарушает нормальное функционирование сетевых сервисов. DoS-атаки также могут пытаться замедлить или остановить системы или сервисы в целевой сети. Существует два типа DoS-атак: шквальная эксплуатация и наводнение (flooding).

D-View. Программное обеспечение SNMP компании D-Link, используемое для управления и мониторинга сетевого оборудования.

E

ESP (англ. Encapsulating Security Payload). Один из протоколов IPSec, который обеспечивает конфиденциальность и целостность протоколов, расположенных выше в стеке протоколов, и дополнительно может обеспечивать анти-replay-сервис, т. е. целостность некоторой последовательности дейтаграмм.

Глоссарий

Ethernet. Самая распространенная на сегодняшний день технология локальных сетей. Описана в семействе стандартов IEEE 802.3. Используется в качестве распределительной системы в сетях 802.11.

F

Filtering. Фильтрация. Процесс проверки пакетов данных в сети и определения адресатов для принятия решения о дальнейшей пересылке (данные локальная сеть, удаленная локальная сеть) или отбрасывании пакета. Фильтрация пакетов выполняется мостами, коммутаторами и маршрутизаторами.

Flow control. Управление потоком. Методы, используемые для контроля над передачей данных между двумя точками сети и позволяющие избегать потери данных в результате переполнения приемных буферов.

Forwarding. Продвижение. Процесс продвижения пакета к месту его назначения посредством сетевого устройства.

Fragmentation. Фрагментация. Функция подуровня MAC, выполняющая дробление исходного кадра на кадры меньшего размера (фрагменты) до его передачи.

Frame. Кадр. Единица информации на канальном уровне сетевой модели OSI. В локальной сети кадр представляет собой единицу данных подуровня MAC, содержащую управляющие данные и пакет сетевого уровня. Иногда для обозначения кадров используется термин «пакет», но термины «кадр» или «фрейм» никогда не используются для обозначения пакетов сетевого уровня. Кадр обычно содержит ограничители, управляющие поля, адреса, контрольную сумму и собственно информацию.

FTP (англ. File Transfer Protocol). Протокол передачи файлов. Протокол FTP относится к протоколам прикладного уровня стека TCP/IP, он предназначен для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер.

G

GRE (англ. Generic Routing Encapsulation). Протокол, позволяющий инкапсулировать один протокол в другой. Регламентируется RFC 1701 и 1702.

H

Hash function. Хеш-функция. Функция, отображающая данные произвольной длины в строку битов фиксированной длины, которая может использоваться для аутентификации исходных данных.

Hash value. Хеш-код. Результат, создаваемый хеш-функцией.

I

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, например, запрашиваемая услуга недоступна, или узел либо маршрутизатор не отвечает. Также на ICMP возлагаются некоторые сервисные функции. Регламентируется RFC 792 и др.

Identification. Идентификация. Сервис, с помощью которого определяются уникальные свойства пользователей, которые позволяют отличать их друг от друга, и способы, с помощью которых пользователи указывают свои идентификации информационной системе. Идентификация тесно связана с аутентификацией.

IEEE (англ. Institute of Electrical and Electronic Engineers). Институт инженеров по электротехнике и радиоэлектронике. Профессиональная организация, основанная в 1963 году для координации разработки компьютерных и коммуникационных стандартов. Институт подготовил группу стандартов 802 для локальных сетей. Членами IEEE являются ANSI и ISO.

IEEE 802.1X authentication. Аутентификация IEEE 802.1X. Аутентификация EAP, использующая в качестве транспорта протокол 802.1X.

IKE (англ. Internet Key Exchange). Стандартный, масштабируемый протокол создания и управления SA (Security Association) в протоколах IPSec.

IMPB (англ. IP-MAC-Port Binding). Функция коммутаторов D-Link, позволяющая контролировать доступ компьютеров в сеть на основе их IP- и MAC-адресов, а также порта подключения.

Integrity. Целостность. Гарантирование того, что информация остается неизменной, корректной и аутентичной. Обеспечение целостности предполагает предотвращение и определение неавторизованного создания, модификации или удаления информации. Примером могут являться меры, гарантирующие, что почтовое сообщение не было изменено при пересылке.

Data integrity. Целостность данных. Сервис безопасности, который обеспечивает невозможность изменения данных без обнаружения этого изменения.

IP (англ. Internet Protocol). IP-протокол. Часть стека протоколов TCP/IP. Описывает программную маршрутизацию пакетов и адресацию устройств. Стандарт используется для передачи через сеть базовых блоков данных и дейтаграмм IP. Обеспечивает передачу пакетов без организации соединений и гарантии доставки. Регламентируется RFC 791 и др.

IP address. IP-адрес. Адрес для протокола IP — 32-битовое (4 байта) значение, определенное в STD 5 (RFC 791) и используемое для представления точек подключения в сети TCP/IP. IP-адрес состоит из номера сети (network portion) и номера узла (host portion) — такое разделение позволяет сделать маршрутизацию более эффективной. Обычно для записи IP-адресов используют десятичную нотацию с разделением точками. Новая версия протокола IPv6

Глоссарий

использует 128-разрядные адреса, позволяющие решить проблему нехватки адресного пространства.

IPsec. (англ. IP Security). Набор протоколов, предназначенный для безопасной передачи пакетов IP через сеть общего пользования.

L

L2TP (англ. Layer Two Tunneling Protocol). Является комбинацией протоколов PPTP и Layer 2 Forwarding (L2F), технологии, предложенной компанией Cisco Systems. В настоящее время существуют две реализации протокола L2TP — L2TP версии 2 (RFC 2661) и L2TP версии 3 (RFC 3931). L2TP версии 2 (L2TPv2) определяет инкапсуляцию кадров PPP и их передачу через IP-сеть. L2TP версии 3 (L2TPv3) определяет инкапсуляцию кадров любых протоколов L2 (PPP, Ethernet, ATM и т. д.) и их передачу через IP-сеть.

LAN (англ. Local Area Network). Локальная сеть. Высокоскоростная компьютерная сеть, покрывающая относительно небольшую площадь. Локальные сети объединяют рабочие станции, периферийные устройства, терминалы и другие устройства, находящиеся в одном здании или на другой небольшой территории.

Latency. Задержка. Временная задержка между моментом, когда устройство получило пакет, и моментом, когда пакет был отправлен на порт назначения.

LCP (англ. Link Control Protocol). Протокол управления каналом стека PPP, позволяющий автоматически устанавливать каналы связи, тестировать их, договариваться об их конфигурации, и снова отключать, когда они не нужны. Дополнительной возможностью LCP является выполнение аутентификации узлов.

M

MAC (англ. Media Access Control). Управление доступом к среде передачи. Подуронен в спецификации IEEE 802. Описывает протоколы, реализующие различные методы доступа к среде передачи, отвечает за физическую адресацию, формирование кадров и обнаружение ошибок.

MAC (англ. Message Authentication Code). Алгоритм, который требует использования секретного ключа. Для вычисления кода аутентификации берутся сообщение переменной длины и секретный ключ. Служит для определения целостности сообщения.

MAC address. MAC-адрес. Стандартный адрес канального уровня, который требуется задавать для каждого порта или устройства, подключенного к локальной сети. Другие устройства используют эти адреса для обнаружения специальных сетевых портов, а также для создания и обновления таблиц маршрутизации и структур данных. Длина MAC-адреса составляет 6 байт, а их содержимое регламентируется IEEE. MAC-адреса также называют аппаратными или физическими адресами.

MIC (англ. Message Integrity Code). Код целостности сообщения. Значение, сгенерированное криптографической функцией.

MPPE (англ. Microsoft Point-to-Point Encryption). Протокол компании Microsoft для шифрования пакетов PPP. Регламентируется RFC 3078.

MTU (англ. Maximum Transmission Unit). Модуль передачи максимального размера. Максимальный размер (в байтах) пакета данных, который можно передать через данный интерфейс.

Multicast. Многоадресная рассылка. Доставка потока данных группе узлов на IP-адрес группы многоадресной рассылки.

Multicast address. Групповой адрес. Общий адрес, который относится к некоторой группе нескольких сетевых устройств.

N

NAT (англ. Network Address Translation). Технология, позволяющая использовать локальную схему адресации хостов, расположенных позади межсетевого экрана, одновременно обеспечивая возможность соединяться этим хостам с внешними ресурсами.

NCP (англ. Network Control Protocols). Семейство протоколов управления сетью стека PPP для установления и конфигурирования различных протоколов сетевого уровня. Для каждого поддерживаемого протокола сетевого уровня должен иметься свой протокол NCP.

NDP (англ. Neighbor Discovery Protocol). Протокол обнаружения соседей. Протокол обмена сообщениями из стека протоколов TCP/IP, используемый совместно с IPv6. Регламентируется RFC 4861.

Network Address. Сетевой адрес. Адрес сетевого уровня, который относится к логическому, а не к физическому сетевому устройству. Он также называется протокольным адресом (protocol address).

Node. Узел. Точка присоединения к сети, устройство, подключенное к сети.

O

Public key. Открытый ключ. Один из ключей алгоритма асимметричного шифрования, который делается доступным всем. С помощью данного ключа осуществляется проверка подписи, если алгоритм асимметричного шифрования используется для подписи, либо осуществляется шифрование, если алгоритм асимметричного шифрования используется для шифрования.

OSPF (англ. Open Shortest Path First). Протокол динамической маршрутизации для IP-сетей. Регламентируется RFC 2328, 5340 и др.

P

Packet. Пакет. Группа битов, включающая данные и служебные поля, представленные в соответствующих форматах, и передаваемая целиком. Структура

Глоссарий

пакета зависит от протокола. В общем случае пакет включает три основных элемента: управляющую информацию (адрес получателя и отправителя, длина пакета и т. п.), передаваемые данные, биты контроля и исправления ошибок.

PAP (англ. Password Authentication Protocol). Простой протокол двухстороннего рукопожатия (2-way handshake). Используется в протоколе PPP для аутентификации. В процесс аутентификации пароль передается в открытом виде.

PPP (англ. Point-to-Point Protocol). Протокол двухточечного соединения, который функционирует на канальном уровне модели OSI и определяет метод транспортировки дейтаграмм различных протоколов сетевого уровня по последовательным каналам типа «точка-точка». Регламентируется RFC 1661, RFC 1662 и др.

PPPoA (англ. Point-to-Point Protocol over AAL5). Протокол канального уровня модели OSI, который описывает инкапсуляцию протокола PPP посредством ATM Adaptation Layer 5 (AAL5). Регламентируется RFC 2364. В основном применяется при подключении в Интернет с использованием кабельных модемов (стандарт DOCSIS) и технологий семейства xDSL.

PPPoE (англ. Point-to-Point Protocol over Ethernet). Обеспечивает способ передачи кадров PPP через Ethernet и позволяет подключать сетевые узлы локальной сети через одно абонентское устройство к концентратору доступа (Access Concentrator, AC), расположенному на стороне провайдера. Регламентируется RFC 2516.

PPTP (англ. Point-to-Point Tunneling Protocol). Протокол туннелирования, который позволяет инкапсулировать кадры PPP в дейтаграммы IP и передавать их через IP-сеть. Регламентируется RFC 2637.

Private key. Закрытый ключ. Один из ключей алгоритма асимметричного шифрования, известный только владельцу. С помощью данного ключа сообщение подписывается, если алгоритм асимметричного шифрования используется для создания подписи, либо осуществляется расшифровывание, если алгоритм асимметричного шифрования используется для шифрования.

Public key certificate. Сертификат открытого ключа. Открытые ключи пользователя вместе с некоторой другой информацией, обработанные некоторым неподделываемым способом с использованием асимметричного шифрования уполномоченным органом сертификации.

Q

QoS (англ. Quality of Service). Качество обслуживания. Показатель эффективности системы передачи данных, который отражает качество передачи.

R

Replay attack. Атаки повторного использования. Пассивный захват данных с последующей их пересылкой целевой системе для получения несанкционированного доступа.

RIP (англ. Routing Information Protocol). Протокол динамической маршрутизации для IP-сетей. Регламентируется RFC 1058, 2453 и др.

RIPng. Протокол RIP для протокола IPv6. Регламентируется RFC 2080.

Router. Маршрутизатор. Устройство сетевого уровня, отвечающее за принятие решений о выборе одного из нескольких путей передачи сетевого трафика. Маршрутизаторы отправляют пакеты из одной сети в другую на основе информации сетевого уровня.

Routing. Маршрутизация. Процесс выбора оптимального пути для передачи сообщения.

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman). Алгоритм асимметричного шифрования, основанный на использовании неразрешимости задачи факторизации. Может использоваться как для создания подписи, так и для шифрования.

S

Secret key. Секретный ключ. Ключ алгоритма симметричного шифрования, который используется как для шифрования, так и для расшифровывания.

Security Gateway. Шлюз безопасности. Маршрутизатор, который реализует IPsec-протоколы.

SNMP (англ. Simple Network Management Protocol). Простой протокол управления сетью. Протокол 7-го уровня модели OSI, который специально разработан для управления и мониторинга сетевых устройств. Протокол SNMP входит в стек протоколов TCP/IP и позволяет получать информацию о состоянии устройств сети, обнаруживать и исправлять неисправности и планировать развитие сети. Регламентируется RFC 1157, 1901-1908, 3411-3418 и др.

SSH (англ. Secure Shell). Безопасная оболочка. Сетевой протокол сеансового уровня, позволяющий производить удаленное управление операционной системой и туннелирование TCP-соединений. Регламентируется RFC 4253 и др.

SSL (англ. Secure Sockets Layer). Протокол, который обеспечивает конфиденциальность и целостность данных прикладного уровня, передаваемых между двумя взаимодействующими приложениями, одно из которых является клиентом, а другое — сервером. Является предшественником протокола TLS.

Глоссарий

STP (англ. Spanning Tree Protocol). Протокол связующего дерева. Стандарт IEEE 802.1D-2004, использующий алгоритм связующего дерева и позволяющий самообучающемуся мосту динамически обрабатывать коммутационные петли в сетевой топологии путем создания связующего дерева. Мосты обнаруживают петли путем обмена сообщениями BPDU с другими мостами и ликвидируют петли посредством блокирования выбранных мостовых интерфейсов.

Symmetric cipher. Симметричное шифрование. Криптосистема, в которой шифрование и расшифровывание выполняются с использованием одного и того же ключа.

Т

Tag. Tag. Идентификационная информация, в том числе и номер.

TCP (англ. Transmission Control Protocol). Протокол управления передачей. Ориентированный на соединение протокол транспортного уровня, обеспечивающий надежную дуплексную передачу данных. TCP входит в стек протоколов TCP/IP. Регламентируется RFC 675, 793, 2581 и другими.

Telnet. Стандартный протокол виртуального терминала из набора протоколов TCP/IP. Протокол Telnet используется для удаленного терминального соединения, что дает возможность пользователям подключаться к удаленным системам и использовать их ресурсы, как если бы они работали через обычный терминал. Регламентируется RFC 15, 854 и др.

TLS (англ. Transport Layer Security). Протокол, который обеспечивает конфиденциальность и целостность данных прикладного уровня, передаваемых между двумя взаимодействующими приложениями, одно из которых является клиентом, а другое — сервером. Регламентируется RFC 2246, 4346 и др.

ToS (англ. Type of Service). Тип сервиса. Поле в заголовке протокола IP, используемое для обеспечения QoS.

Trunk. Магистраль. Физическое и логическое соединение между двумя коммутаторами, по которому передается сетевой трафик.

У

UDP (англ. User Datagram Protocol). Протокол дейтаграмм пользователя. Протокол транспортного уровня, не требующий подтверждения соединения. Входит в стек протоколов TCP/IP. UDP обеспечивает обмен дейтаграммами без подтверждения и гарантий доставки.

В

VLAN (англ. Virtual LAN). Виртуальная локальная сеть. Группа устройств, принадлежащих одной или нескольким локальным сетям и сконфигуриро-

ванных таким образом (с помощью программного обеспечения), что обмен данными между ними происходит так, как будто они подключены к одному кабелю, хотя на самом деле находятся в разных сегментах локальной сети. VLAN основаны на логическом соединении.

VPN (англ. Virtual Private Network). Виртуальные локальные сети. Различные технологии, которые позволяют создавать логические сети, использующие в качестве транспорта другие сетевые протоколы. При этом характеристики безопасности созданной логической сети могут отличаться от характеристик безопасности сети, которая используется в качестве транспорта.

Учебное издание

Компьютерные системы и сети

**Смирнова Елена Викторовна
Пролетарский Андрей Викторович
Ромашкина Екатерина Александровна**

Технологии TCP/IP в современных компьютерных сетях

Оригинал-макет подготовлен
в Издательстве МГТУ им. Н.Э. Баумана.

В оформлении использованы шрифты
Студии Артемия Лебедева.

Подписано в печать 12.04.2019. Формат 70×100/16.
Усл. печ. л. 35,75. Тираж 1000 экз. Заказ

Издательство МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
press@bmstu.ru
www.baumanpress.ru