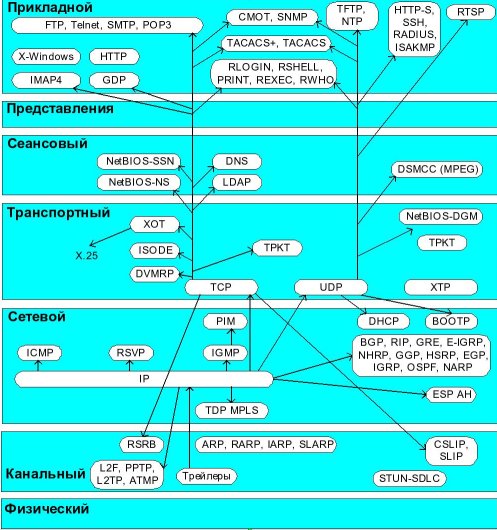
Стек протоколов TCP/IP  
краткий отзор структуры стека протоколов TCP/IP, связи между протоколами стека  


[PDF](http://www.protocols.ru/files/Protocols/TCPIP.pdf)

Агентство DARPA (Defense Advance Research Projects Agency) разработало протоколы TCP/IP (Transmission Control Protocol/Internet Protocol) для объединения в сеть компьютеров различных подразделений министерства обороны США. Международная распределенная сеть Internet использует стек протоколов TCP/IP для объединения компьютерных ресурсов всей планеты. Достаточно часто эти протоколы используются также в частных и коммерческих сетях. Стек TCP/IP включает следующие протоколы:

* [IP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=21) /[IPv6](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=22) - Internet Protocol.
* [TCP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=23) - Transmission Control Protocol.
* [UDP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=24) - User Datagram Protocol.

***Канальный уровень***

* [ARP/RARP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=25) - Address Resolution Protocol/Reverse Address.

*Протоколы туннелирования*

* [ATMP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=26) - Ascend Tunnel Management Protocol.
* [L2F](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=27) - Layer 2 Forwarding Protocol.
* [L2TP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=28) - Layer 2 Tunneling Protocol.
* [PPTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=29) - Point-to-Point Tunneling Protocol.

***Сетевой уровень***

* [DHCP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=30) / [DHCPv6](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=31) - Dynamic Host Configuration Protocol.
* [DVMRP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=32) - Distance Vector Multicast Routing Protocol.
* [ICMP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=33) /[ICMPv6](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=34) - Internet Control Message Protocol.
* [IGMP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=35) - Internet Group Management Protocol.
* [MARS](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=36) - Multicast Address Resolution Server.
* [PIM](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=37) - Protocol Independent Mulyicast.
* [RIP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=38) - Routing Information Protocol.
* RIP2 - Routing Information Protocol II.
* [RIPng for IPv6.](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=39)
* [RSVP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=40) - Resource ReSerVation setup Protocol.

*Безопасность*

* [AH](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=41) - Authentication Header.
* [ESP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=42) - Encapsulating Security Payload.

*Маршрутизация*

* [BGP-4](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=43) - Border Gateway Protocol.
* [EGP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=44) - Exterior Gateway Protocol.
* [EIGRP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=45) - Enhanced Interior Gateway Routing Protocol.
* [GRE](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=46) - Generic Routing Encapsulation.
* [HSRP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=47) - Cisco Hot Standby Router Protocol.
* [IGRP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=48) - Interior Gateway Routing.
* [NARP](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=49) - NBMA Address Resolution Protocol
* NHRP - Next Hop Resolution Protocol.
* [OSPF](http://www3.protocols.ru/modules.php?name=Content&pa=showpage&pid=50) - Open Shortest Path First.

***Транспортный уровень***

* [Mobile IP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=52) .
* [Van Jacobson](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=53) - compressed TCP.
* [XOT](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=54) - X.25 over TCP.

*VoIP*

* [MGCP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=55) - Media Gateway Control Protocol.
* [SGCP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=56) - Simple Gateway Control Protocol.

***Сеансовый уровень***

* [DNS](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=57) - Domain Name Service.
* [NetBIOS/IP.](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=58)

***Прикладной уровень***

* [FTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=59) - File Transfer Protocol.
* [Finger](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=61) User Information Protocol.
* [TFTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=60) - Trivial File Transfer Protocol.
* [Gopher](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=62) - Internet Gopher Protocol.
* [HTTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=63) - Hypertext Transfer Protocol.
* [S-HTTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=64) - Secure Hypertext Transfer Protocol.
* [IMAP4](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=65) - Internet Message Access Protocol rev 4.
* [IPDC](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=66) - IP Device Control.
* [ISAKMP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=67) - Internet Message Access Protocol version 4rev1.
* [NTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=68) - Network Time Protocol.
* [POP3](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=69) - Post Office Protocol version 3.
* [Radius](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=70) .
* [RLOGIN](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=71) - Remote Login.
* [RTSP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=72) - Real-time Streaming Protocol.
* [SMTP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=73) - Simple Mail Transfer Protocol.
* [SNMP](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=74) - Simple Network Management Protocol.
* [TACACS+](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=75) - Terminal Access Controller Access Control System.
* [TELNET](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=76) .
* [X-Window](http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=77) .

Положение протоколов стека TCP/IP в модели OSI показано на рисунке.

# IP

RFC 791 [*http://www.cis.ohio-state.edu/htbin/rfc/rfc791.html*](http://www.cis.ohio-state.edu/htbin/rfc/rfc791.html) (перевод на русский язык - [*PDF*](http://www.protocols.ru/files/RFC/rfc791.pdf) )

RFC 1853 [*http://www.cis.ohio-state.edu/htbin/rfc/rfc1853.html*](http://www.cis.ohio-state.edu/htbin/rfc/rfc791.html)

IP (Internet Protocol) представляет собой протокол уровня маршрутизируемых дейтаграмм в стеке TCP/IP. Все другие протоколы стека TCP/IP (кроме ARP и RARP) используют протокол IP для маршрутизации кадров между хостами. Заголовок кадров IP содержит маршрутную и управляющую информацию, связанную с доставкой дейтаграмм.

Структура заголовков IP показана на рисунке.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 8 | 16 | 32 | |
| Версия | IHL | Тип сервиса | Общий размер | |
| Идентификация | | | Флаги | Смещение фрагмента |
| Время жизни | | Протокол | Контрольная сумма заголовка | |
| Адрес отправителя | | | | |
| Адрес получателя | | | | |
| Опции и заполнение | | | | |
| Данные | | | | |

Структура заголовка IP.

##### Версия

Поле версии определяет формат заголовка Internet.

##### IHL

Internet Header Length - размер заголовка Internet указывает размер заголовка в 32-битовых словах, задавая смещение данных от начала пакета. Минимальный размер заголовка составляет 5 слов (160 битов).

##### Тип сервиса

Показывает желаемый уровень качества обслуживания. Сети могут обеспечивать различный уровень преимуществ при доставке, играющий важную роль при условиях высокой загрузки сети. Поддерживаются также три опции качества обслуживания - малая задержка, высокая надежность и высокая пропускная способность.

Биты 0 - 2 - преимущественная доставка

111 сетевое управление

110 межсетевое управление

101 CRITIC/ECP

100 Flash override

011 Flash

010 немедленная доставка

001 приоритетная доставка

000 Routine (нормальный режим)

Бит 3 - задержка

0 Нормальная

1 Малая

Бит 4 - пропускная способность

0 Нормальная

1 Высокая

Бит 5 - надежность доставки

0 Нормальная

1 Высокая

Биты 6 - 7 - зарезервированы для использования в будущем

##### Общий размер

Размер дейтаграммы в байтах с учетом заголовка и данных. Размер поля позволяет использовать дейтаграммы длиной до 65535 байтов, хотя такой размер нежелателен для многих сетей и хостов. Все хосты должны быть готовы к приему дейтаграмм размером до 576 байтов, независимо от того как они доставляются - целиком или фрагментами. Рекомендуется передавать дейтаграммы, размер которых превышает 576 байтов только в тех случаях, когда адресат готов принять такие дейтаграммы.

##### Идентификация

Значение идентификатора, которое отправитель задает для обеспечения корректного порядка сборки фрагментов дейтаграммы на приемной стороне.

##### Флаги

Трехбитовое поле флагов управления:

Бит 0 - зарезервирован и должен иметь нулевое значение

Бит 1 - возможность фрагментирования

0 Можно фрагментировать

1 Не фрагментировать

Бит 2 - наличие дополнительных фрагментов

0 Последний фрагмент

1 Есть последующие фрагменты

##### Смещение фрагмента

13-битовое значение, задающее смещение фрагмента от начала целой дейтаграммы. Смещение фрагмента измеряется в 8-байтовых (64 бита) словах. Первый фрагмент имеет нулевое смещение.

##### Время жизни

Показывает максимальное время существования дейтаграммы в сети Internet. При нулевом значении этого поля дейтаграмма должна быть уничтожена. Время жизни дейтаграмм измеряется в секундах. Однако, поскольку каждый модуль, работающий с дейтаграммой, должен уменьшать значение поля TTL (time-to-life) по крайней мере на 1 (даже в тех случаях, когда обработка дейтаграммы занимает меньше секунды), значение этого поля должно быть не меньше желаемого времени жизни дейтаграммы. Дейтаграммы с истекшим в процессе доставки временем жизни не попадают к получателю.

##### Протокол

Указывает протокол следующего уровня, содержащийся в поле данных дейтаграммы IP.

##### Контрольная сумма заголовка

Контрольная сумма, рассчитанная с учетом только полей заголовка дейтаграммы. Поскольку некоторые поля заголовка (например, время жизни) могут меняться в процессе доставки, значение контрольной суммы заново вычисляется и проверяется в каждой точке обработки заголовков.

##### Адрес отправителя/ получателя

32-битовые значения адресов отправителя и получателя дейтаграммы. Следует четко различать имена, адреса и маршруты. Имя показывает название объекта, адрес говорит о его местоположении в сети, а маршрут - показывает путь к объекту. Протокол IP имеет дело преимущественно с адресами. Связь между адресами и именами реализуется протоколами вышележащих уровней. Модуль Internet отображает адреса IP на локальные сетевые адреса. Связь локальных сетевых адресов с маршрутами обеспечивается протоколами нижележащих уровней.

##### Опции

Это поле содержит необязательные опции дейтаграммы. Используемые опции должны быть реализованы во всех модулях IP (хосты и шлюзы). В некоторых опции безопасности являются обязательными для всех дейтаграмм.

Поле опций имеет переменную длину и может содержать различное число опций. Существуют два формата опций:

* Однооктетные опции
* Многооктетные опции, содержащие поля типа опции (1 октет), ее размера (1 октет) и собственно опций.

Поле длины опции учитывает все субполя опции - тип, размер и сами опции.

Октет типа опции имеет три поля:

1 бит - флаг копирования показывает, что должна ли данная опция копироваться во все фрагменты дейтаграммы:

0 опция копируется

1 опция не копируется.

2 бита - класс опции:

0 управление

1 зарезервировано

2 отладка и измерение

3 зарезервировано

5 битов - номер опции

##### Данные

Данные IP или протоколов вышележащих уровней.

<http://www.protocols.ru/modules.php?name=Content&pa=showpage&pid=21>

# Протоколы семейства TCP/IP

|  |  |  |  |
| --- | --- | --- | --- |
| **Протоколы** | | | |
| [IP](http://lib.ru/LABIRINT/protocol.htm#IP) | [RIP](http://lib.ru/LABIRINT/protocol.htm#RIP) | [UDP](http://lib.ru/LABIRINT/protocol.htm#UDP) | [TCP](http://lib.ru/LABIRINT/protocol.htm#TCP) |

**Межсетевой протокол IP**. Модуль IP является базовым элементом технологии Internet. Его центральной частью является таблица маршрутов. Таблица маршрутов заполняется администратором сети и обычно инициализируется в момент загрузки системы. Когда речь идет о простой локальной IP-сети, то протокол IP мало что добавляет к услугам Ethernet, за исключением того, что в сети будут работать все прикладные программы, реализованные для IP-технологии. Однако ситуация меняется, если речь идет о сетях, сопряженных шлюзом.

**Протокол RIP (Routing Information Protocol)**. Протокол предназначен для автоматического обновления таблицы маршрутов. При этом используется информация о состоянии сети, которая рассылается маршрутизаторами (routers). В соответствии с протоколом RIP любая машина может быть маршрутизатором. При этом все маршрутизаторы делятся на активные и пассивные. Активные маршрутизаторы сообщают о маршрутах, которые они поддерживают в сети. Пассивные маршрутизаторы читают эти широковещательные сообщения и исправляют свои таблицы маршрутов, но при этом сами информации в сеть не предоставляют. Обычно в качестве активных маршрутизаторов выступают шлюзы, а в качестве пассивных - обычные машины (hosts).

**Протокол UDP**. Этот протокол является одним из двух основных транспортных протоколов, расположенных сразу над IP. К заголовку IP-пакета UDP добавляет два поля: порт и контрольная сумма. Поле “порт” позволяет мультиплексировать информацию между разными прикладными процессами. Поле "контрольная сумма" позволяет поддерживать целостность данных

**Протокол TCP**. Предоставляет другой способ доставки сообщений, отличный от UDP. Вместо "ненадежной" доставки датаграмм без установления соединения, TCP обеспечивает гарантированную доставку с установлением соединения в виде байтовых потоков.

Прикладные программы взаимодействуют с модулем TCP также через порты. Существуют определенные стандартом номера портов, которые отведены под обслуживание стандартных сервисов Internet. Так telnet обслуживается через 23 порт, почта (SMTP) - через 25 и т.п.

Когда два процесса начинают общаться через модули TCP, то эти модули поддерживают информацию о состоянии соединения, которое называется виртуальным каналом. Канал является дуплексным, т.е. информация может передаваться одновременно в двух направлениях.

Согласно протоколу TCP, поток байтов разбивается на пакеты. Любые данные для модуля TCP представляются в виде потока байтов. На другом конце виртуального канала данные снова собираются в поток. Модуль TCP не сохраняет разделения потоков данных на записи. Так можно записать в канал 5 записей по 80 байт, а прочитать одну в 400 байтов длиной

# Подробное описание протокола TCP

TCP (Transmission Control Protocol, Протокол управления передачей) был спроектирован в качестве связующего протокола для обеспечения интерактивной работы между компьютерами. TCP обеспечивает надежность и достоверность обмена данными между процессами на компьютерах, входящих в общую сеть. TCP, с одной стороны, взаимодействует с прикладным протоколом пользовательского приложения, а с другой, с протоколом, обеспечивающим "низкоуровневые" функции: маршрутизацию и адресацию пакетов, которые, как правило, выполняет IP.

В операционной системе реализация TCP представляет собой отдельный системный модуль (драйвер), через который, как правило, проходят все вызовы функций протокола. Интерфейс между прикладным процессом и TCP представляет собой библиотеку вызовов, такую же как библиотека системных вызовов, например, для работы с файлами. Вы можете открыть или закрыть соединение (как открыть или закрыть файл) и отправить или принять данные из установленного соединения (аналогично операциям чтения и записи файла). Вызовы TCP могут работать с прикладным приложением в асинхронном режиме. Безусловно, реализация TCP в каждой системе может осуществлять множество собственных функций, но любая из этих реализации должна обеспечивать минимум функциональности, которая требуется стандартами TCP.

Схема работы пользовательского приложения с TCP, в общих чертах, состоит в следующем. Для передачи данных пользовательскому процессу надо вызвать соответствующую функцию TCP, с указанием на буфер передаваемых данных. TCP упаковывает эти данные в сегменты своего стека и вызывает функцию передачи протокола нижнего уровня, например IP.

На другом конце, получатель TCP группирует поступившие от протокола нижнего уровня данные в принимающие сегменты своего буфера, проверяет целостность данных, передает данные пользовательскому процессу и уведомляет отправителя об их получении.

Пользовательский интерфейс с TCP может выполнять такие команды как открыть (OPEN) или закрыть (CLOSE) соединение, отправить (SEND) или принять (RECEIVE) данные, или получить статус соединения (STATUS). Эти вызовы подобны любым другим вызовам функций операционной системы из пользовательской программы, таким как открытие, чтение или закрытие файла.

В модели межсетевого соединения взаимодействие TCP и протоколов нижнего уровня, как правило, не специфицировано, за исключением того, что должен существовать механизм, который обеспечивал бы асинхронную передачу информации от одного уровня к другому. Результатом работы этого механизма является инкапсуляция протокола более высокого уровня в тело протокола более низкого уровня. Реализуется этот механизм через интерфейс вызовов между TCP и IP.

В результате работы этого механизма каждый TCP-пакет вкладывается в "конверт" протокола нижнего уровня, например, IP. Получившаяся таким образом дейтаграмма содержит в себе TCP-пакет так же как TCP-пакет содержит пользовательские данные.

Простейшая модель работы TCP-протокола выглядит обманчиво гладко, поскольку на самом деле реальная работа изобилует множеством деталей и тонкостей.

Логическая структура сетевого программного обеспечения, реализующего протоколы семейства TCP/IP в каждом узле сети Internet, изображена на рис. 2.12.

Прямоугольники обозначают обработку данных, а линии, соединяющие прямоугольники, - пути передачи данных. Горизонтальная линия внизу рисунка обозначает кабель сети Ethernet, которая используется в качестве примера физической среды. Понимание этой логической структуры является основой для понимания всей технологии TCP/IP.

Рис. 2.12.Структура сетевого программного обеспечения семейства протоколов TCP/IP

Далее более подробно рассмотрим возможности, принципы построения и основные функции протокола TCP:

 Потоки данных, стек протоколов, механизм гнезд и мультиплексирование соединений

 Процедура установления соединения и передача данных

 Механизмы обеспечения достоверности передаваемых данных

 Механизм контроля потока данных

 Флаг важности пакета, средства обеспечения безопасности протокола

### Потоки данных, стек протоколов, механизм гнезд и мультиплексирование соединений

Для установления соединения между двумя процессами на различных компьютерах сети необходимо знать не только Internet-адреса компьютеров, но и номер ТСР-порта, который процесс использует на данном компьютере. В совокупности с Internet-адресом компьютера порты образуют систему гнезд (sockets). Пара гнезд уникально идентифицирует каждое соединение или поток данных в сети Internet, а порт обеспечивает независимость каждого ТСР-канала на данном компьютере. Безусловно, несколько процессов на машине могут использовать один и тот же ТСР-порт, но с точки зрения удаленного процесса между этими процессами не будет никакой разницы.

Рассмотрим потоки данных, проходящие через протоколы. При использовании протокола TCP данные передаются между прикладным процессом и модулем TCP. Типичным прикладным процессом, использующим протокол TCP, является модуль FTP (File Transfer Protocol, Протокол передачи фай-лов). Стек протоколов в этом случае будет FTP/TCP/IP/ENET. При использовании протокола UDP (User Datagram Protocol, Протокол дейтаграмм пользователя) данные передаются между прикладным процессом и модулем UDP. Например, SNMP (Simple Network Management Protocol, Простой протокол управления сетью) пользуется транспортными услугами UDP. Его стек протоколов выглядит так: SNMP/UDP/IP/ENET.

Одно гнездо на компьютере может быть задействовано в соединениях с несколькими гнездами на удаленных компьютерах. Кроме того, одно и то же гнездо может передавать поток данных в обоих направлениях. Таким образом, механизм гнезд позволяет на одном компьютере одновременно работать нескольким приложениям и уникально идентифицирует каждый поток данных сети. Это называется мультиплексированием соединений.

Модули TCP, UDP и драйвер Ethernet являются мультиплексорами типа n x 1. Действуя как мультиплексоры, они переключают несколько входов на один выход. Они также являются демультиплексорами типа 1 х n. Как демультиплексоры, они переключают один вход на один из многих выходов в соответствии с полем типа в заголовке протокольного блока данных. Когда Ethernet-кадр попадает в драйвер сетевого интерфейса Ethernet, он может быть направлен либо в модуль ARP, либо в модуль IP. (Значение поля типа в заголовке кадра указывает, куда должен быть направлен Ethernet-кадр).

Если IP-пакет попадает в модуль IP, то содержащиеся в нем данные могут быть переданы либо модулю TCP, либо UDP, что определяется полем "Protocol" в заголовке IP-пакета. Если TCP-сообщение попадает в модуль TCP, то выбор прикладной программы, которой должно быть передано сообщение, осуществляется на основе значения поля "порт" в заголовке TCP-сообщения.

Мультиплексирование данных в обратную сторону осуществляется довольно просто, так как из каждого модуля существует только один путь вниз. Каждый протокольный модуль добавляет к пакету свой заголовок, на основании которого машина, принявшая пакет, выполняет демультиплексирование.

Назначение портов приложениям на каждом компьютере происходит независимо друг от друга. TCP может самостоятельно выбирать порт, с которым будет работать приложение, или приложение укажет, с каким портом на данном компьютере оно будет работать. Однако, как правило, часто используемые приложения - сервисы используют одни и те же номера портов, которые уже стали общеизвестными, например, такие как HTTP, FTP, SMTP и др., для того, чтобы к данному процессу на компьютере можно было присоединиться, указывая только адрес машины. Например, Internet браузер, если ему не указать дополнительно, ищет по указанному адресу приложение, работающее с портом 80, - это наиболее распространенный порт для серверов WWW.

Кроме того, машина может быть снабжена несколькими сетевыми интерфейсами, тогда она должна осуществлять мультиплексирование типа п х т, т. е. между несколькими прикладными программами и сетевыми интерфейсами.

### Установление соединения и передача данных

Соединение определяется вызовом команды OPEN с аргументами в виде номера локального порта и гнезда (IP-адрес + порт) удаленного процесса. Функция OPEN вызывается и в том случае, когда данный процесс намерен передавать информацию (активный OPEN), и когда процесс ожидает поступления информации (пассивный OPEN). Функция возвращает идентификатор соединения, по которому пользователь может ссылаться в своих последующих вызовах. Идентификатор соединения указывает на структуру данных, в которой хранятся переменные и информация данного TCP-соединения. Эта структура данных называется Управляющая Структура Передачи - Transmission Control Block (TCB).

ТСВ хранит такие параметры соединения, как адреса локального и удаленного гнезд, указатели на полученные и отправляемые пользовательские данные, указатели на очередь блоков для повторной отправки, номер текущего сегмента и т. д., то есть всю информацию, используемую данным соединением.

Как уже отмечено выше, открытие соединения может быть активным и пассивным. Пассивное открытие обозначает, что процесс ожидает поступления сигнала открытия соединения и не пытается открыть канал самостоятельно. Иными словами, канал, открытый с одного конца как пассивный, ожидает инициирующего сигнала от какого-либо из хостов. Этот тип открытия канала используется процессами, которые предоставляют свой сервис через заранее известный номер своего порта (например, HTTP, SMTP и т. д.) и работают с механизмом общеизвестных гнезд.

Процесс может вызвать функцию пассивного открытия канала и ждать получения сигнала активного открытия канала от другого процесса, и только после получения такого сигнала соединение будет установлено. Соединение будет так же установлено, если два процесса активно откроют канал навстречу друг другу. Эта гибкость в установлении соединения особенно важна в распределенных сетях, когда компьютеры работают асинхронно.

Только в двух случаях принципиально важно, чтобы гнездо на данном компьютере было открыто как пассивное, а на другой стороне как активное:   
1. Когда локальное пассивное открытие соединения полностью определяет гнездо на другой стороне.   
2. Локальное пассивное открытие гнезда не предполагает каких-либо ограничений на гнездо с другой стороны, т. е. любое гнездо может присоединиться к данному сервису.

Если у вас на одном гнезде открыто пассивно, с соответствующими записями ТСВ, несколько соединений, то при открытии активного соединения на другом конце будет задействовано ТСВ, соответствующее параметрам, заданным активным гнездом. Если такого не существует, выбирается любое другое открытое гнездо.

Процедура установления соединения использует специальный флаг синхронизации - SYN и состоит из трех этапов обмена сообщениями - так называемое "трехходовое квитирование". Использование именно трех тактов квитирующих сообщений всегда достаточно, чтобы синхронизировать потоки данных.

Инициализация соединения начинается с обмена пакетами, которые отправляются при открытии канала пользователем, например, командой OPEN, и содержат флаг SYN и свой начальный порядковый номер пакетов данных. После этого гнезда ожидают пакет с ТСВ-партнера. Соединение считается установленным, когда в обоих направлениях синхронизируются нумерующие последовательности передаваемых пакетов, т. е. и клиент и сервер "знают", пакет с каким номером поступит с противоположного конца соединения.

Соединение закрывается, когда гнезда обмениваются пакетами, содержащими команду FIN. При этом все ресурсы системы, занятые ТСВ-данных соединений, должны быть освобождены.

Вообще говоря, TCP сам определяет, как группировать и когда отправлять очередной блок данных. Однако в некоторых случаях, пользователю необходимо быть уверенным, что все данные, переданные на уровень TCP, отправлены. Для этих целей существует функция "проталкивания пакета" - PUSH-функция. Вызов этой функции позволяет проконтролировать отправку всех буферизированных TCP-протоколом данных.

Примечание:  
Под словом "пользователь" здесь и далее будет пониматься пользовательский процесс или приложение, которое работает с TCP.

Пользователь, отправляющий пакет вызовом SEND, указывает, передаются ли в сегменте или в предшествующих сегментах данные немедленно, или при передаче будет использоваться механизм кэширования данных, тогда данные будут отправляться в соответствии с установками системы. Когда же TCP получает пакет с флагом PUSH, а этот флаг устанавливается PUSH-функцией, вся кэшированная информация, вне зависимости от того, сформирована она одним вызовом команды SEND или несколькими, будет немедленно отправлена получателю.

Назначение PUSH-функции и PUSH-флага состоит только в "проталкивании" данных к пользователю, минуя механизм кэширования. Команда PUSH не производит никаких дополнительных группировок или других действий над данными.

Следует отметить, что существует взаимосвязь между PUSH-функцией и использованием буферов данных, используемых как TCP, так и пользовательским интерфейсом. Всякий раз, когда с данными, помещенными в пользовательский буфер получателя, ассоциирован PUSH-флаг, буфер немедленно передается пользователю, даже если он не заполнен до конца. Если данные заполнили буфер до получения PUSH-флага, эти данные передадутся пользователю в размере целого буфера.

### Механизмы обеспечения достоверности передаваемых данных

Протокол ТСР должен уметь работать с поврежденными, потерянными, дублированными или поступившими с изменением порядка пакетами. Это достигается благодаря механизму присвоения порядкового номера каждому передаваемому пакету данных и механизму проверок получения пакетов подтверждения доставки.

Когда TCP передает сегмент данных, копия этих данных помещается в очередь повтора передачи и запускается таймер ожидания подтверждения. Когда система получает подтверждение - сегмент TCP, содержащий управляющий флаг АСК, что этот пакет данных получен, она удаляет его из очереди. Сегмент подтверждения получения содержит номер полученного сегмента, на основании которого и происходит контроль доставки данных адресату. Если подтверждение не поступило до истечения срока таймера, пакет отправляется еще раз. Уведомление TCP о получении пакета данных еще не означает, что он был доставлен конечному пользователю. Оно только означает, что TCP выполнил возложенные на него функции.

При передаче информации каждому байту данных присваивается порядковый номер, поэтому, в какой бы последовательности эти пакеты не достигали точки назначения, они всегда будут собраны в изначально заданной последовательности. Порядковый номер первого байта данных в передаваемом сегменте называется порядковым номером сегмента. Нумерация проводится "с головы состава", т. е. от заголовка пакета. TCP-пакет также содержит "подтверждающий номер" (acknowledgment number), который представляет собой номер следующего ожидаемого пакета данных передачи в обратном направлении. Иными словами, номер обозначает: "до сих пор я все получил". Механизм с использованием "подтверждающего номера" позволяет исключать дублирование пакетов при повторной отправке недоставленных данных.

Кроме определения порядка следования информационных пакетов, "порядковый номер" играет большую роль в механизме синхронизации соединения и контроле потерянных пакетов при разрывах соединения. Однако необходимо помнить, что величина счетчика - нумератора все же ограничена. Пакеты могут нумероваться числами от 0 до 2(32-1). Таким образом, все арифметические операции со счетчиком пакетов производятся по модулю 232. Это не означает, что гнезда, в процессе соединения, могут обмениваться только ограниченным количеством пакетов. Поскольку в процессе обмена получатель и отправитель знают предыдущий, последующий номера пакетов и длину пакета, а эти величины хранятся в структуре ТСВ при образовании соединения, все операции сравнения по модулю 232 проводятся корректно.

Здесь стоит сказать несколько слов о механизме предотвращения появления в сети пакетов с одинаковыми номерами. Они могут появиться, например, при установлении и быстром сбросе соединения или при сбросе соединения и его быстром восстановлении, т. е. когда номер испорченного пакета может сразу использоваться новым пакетом данных. Механизм предотвращения подобных ситуаций основан на генерировании начального числа последовательности пакетов, а поскольку счетчик циклический, то не все ли равно, с какого места начинать отсчет.

Так, при установлении нового соединения генерируется 32-битное число ISN (Initial Sequence Number). Генератор использует 32 младших разряда машинного таймера, который меняется каждые 4 микросекунды (полный цикл - 4,55 часа). Это число и служит отсчетом нумератора пакетов. Кроме того, каждая дейтаграмма в сети имеет ограниченное время жизни MSL - Maximum Life Time, которое значительно меньше периода генератора. Таким образом, в сети гарантируется невозможность возникновения конфликтов пакетов с одинаковыми номерами.

Поврежденные пакеты отсеиваются механизмом проверки величины контрольной суммы данных, которая размещается в каждом передаваемом пакете.

### Механизм контроля потока данных

Протокол TCP позволяет получателю последовательности пакетов регулировать передаваемый отправителем поток данных. Этот механизм основан на том, что при передаче флага подтверждения получения пакета (АСК) в ТСР-сегменте передается размер буфера данных, который может быть передан отправителем, не дожидаясь разрешения на отправку очередной порции данных от получателя. Иными словами - передается размер свободного места в стеке протокола, куда записываются только что принятые и ожидающие дальнейшей обработки данные. Из стека данные обрабатываются и передаются соответствующим процессам. Этот механизм позволяет избегать "пробок" при передаче данных между системами различной производительности.

"Окно" задается в количестве байт, отсчитываемых от номера байта, заданного в поле номера пакета "подтвержденного получения" (acknowledgment number) данных. Нулевой размер окна означает для отправителя команду приостановить передачу до готовности принимать данные получателем. Необходимо заметить, что в этом случае отправитель посылает однобайтные пакеты и на основании информации пакетов подтверждения возобновляет или нет дальнейшую передачу данных.

Безусловно, большой размер окна позволяет передавать данные быстрее, поскольку отправителю пакета не нужно ждать сигнала готовности приема от получателя. Однако в случае сбоя передачи, соответственно, возрастет объем данных, которые нужно отправить заново. При небольшом же размере окна сбойные сегменты данных можно локализовать с минимальными потерями.

Механизм контроля потока данных позволяет TCP влиять на изменение размера "окна" передачи. Это, в свою очередь, помогает значительно оптимизировать скорость достоверного обмена данных между процессами в гетерогенных сетях Internet.

### Флаг важности пакета, средства обеспечения безопасности протокола

Протокол TCP наследует все параметры секретности и приоритетности пакета, используемые IP и позволяет работать с ними для обеспечения собственной безопасности. В TCP-спецификации термин "секретность и изолированность" обозначает безопасность соединения, изолированность потоков данных и другие ограничения, которые обеспечивает IP.

Кроме того, TCP поддерживает некоторые дополнительные механизмы, которые поддерживаются специальным аппаратным и программным обеспечением TCP.

Для того чтобы TCP мог полностью реализовать свои возможности работы с иерархией приоритетных пакетов и информацией различного уровня секретности, необходима также поддержка этих возможностей и со стороны протоколов высокого уровня, и самих пользовательских приложений, и операционных систем.

Кроме наследуемых от IP-параметров, TCP имеет в своем распоряжении механизм работы с флагом важности пакетов - URG. Этот механизм позволяет отправителю "настоятельно сообщать" получателю о том, что тот или иной пакет содержит срочную информацию и позволяет получателю сигнализировать, когда вся срочная информация им получена. Такой механизм уведомления, т. е. манипулирование с URG-флагом TCP-пакета используется, например, при обработке асинхронных событий.

### Состав и предназначение полей заголовка

ТСР-сегменты отправляются как IP-дейтаграммы. Заголовок TCP, следующий за IP-заголовком, содержит информацию TCP-протокола (рис. 2.13).

**Source Port** (16 бит). Порт отправителя.

**Destination Port** (16 бит). Порт получателя.

**Sequence Number** (32 бита). Номер кадра. Номер кадра первого октета данных в этом сегменте (за исключением пакета, где присутствует флаг SYN). Если в пакете присутствует флаг SYN, то номер данного пакета становится номером начала последовательности (ISN) и номером первого октета данных становится номер ISN+1.

Рис. 2.13. Заголовок TCP-пакета

**Acknowledgment Number** (32 бита). Поле номера кадра подтвержденного получения. Если пакет содержит установленный контрольный бит АСК, то это поле содержит номер следующего пакета данных отправителя, который ожидает получатель. При установленном соединении пакет подтверждения отправляется всегда.

**Data Offset** (4 бита). Поле величины смещения данных. Оно содержит количество 32-битных слов заголовка TCP-пакета. Это число определяет смещение расположения данных в пакете.

**Reserved** (6 бит). Резервное поле. Поле зарезервировано.

Флаги управления (слева направо):

URG: Флаг срочности   
АСК: Флаг пакета, содержащего подтверждение получения   
PSH: Флаг форсированной отправки   
RST: Переустановка соединения   
SYN: Синхронизация чисел последовательности   
FIN: Флаг окончания передачи со стороны отправителя

**Window** (16 бит). Окно. Это поле содержит количество байт данных, которое отправитель данного сегмента может принять, отсчитанное от номера байта, указанного в поле Acknowledgment Number.

**Checksum** (16 бит). Поле контрольной суммы. Это поле содержит 16 бит суммы побитных дополнений 16-битных слов заголовка и данных. Если сегмент содержит нечетное число байт заголовка и данных, последний байт дополняется справа нулями. При вычислении контрольной суммы поле контрольной суммы полагается равным нулю.

**Urgent Pointer** (16 бит). Поле указателя срочных данных. Это поле содержит значение счетчика пакетов, начиная с которого следуют пакеты повышенной срочности. Это поле принимается во внимание только в сегментах с установленным флагом URG.

**Options**. Поле дополнительных параметров: может быть переменной длины.

**Padding.** Заполнение: переменная длина. Заполнение (нулями) TCP-заголовка используется для выравнивания его по 32-битному слову.

### Псевдозаголовок

Между TCP-заголовком и IP-заголовком располагается так называемый псевдозаголовок (рис. 2.14). Он состоит из IP-адреса отправителя (Source Address), IP-адреса получателя (Destination Address), типа протокола (PTCL) и длины TCP-пакета (TCP Length). Он предназначен для "страховки" неправильной маршрутизации TCP-пакета. Информация, расположенная в этом заголовке передается TCP от IP-протокола как аргумент или результат обработки вызова IP-функции.

Рис. 2.14. Формат псевдозаголовка   
Более подробное описание протокола TCP можно найти в RFC-793, RFC-1180.

--------------------------------------------------

##### Использован источник C. Золотов "Протоколы Internet" 1998г. без изменений.

**1.1 Принципы построения и организационная структура Интернет.**

**1.1.1 Каким образом мы попадаем со своего компьютера на удаленный сервер?**

Маршрутизация в сети Internet - **пакетная** (бывает еще канальная, как в телефонии, АТС - маршрутизатор).

Все наши компьютеры объединены в локальную сеть, и имеют локальную IP-адресацию. Пакеты с такой адресацией "путешествовать" в глобальной сети не смогут, т.к. маршрутизаторы их не пропустят.

Поэтому существует шлюз, который преобразовывает пакеты с локальными IP-адресами, давая им свой внешний адрес. И дальше ваши пакеты путешествуют с адресом шлюза.

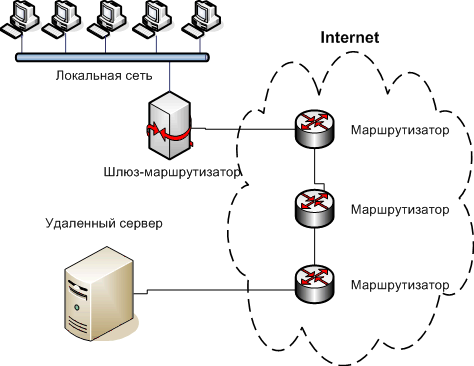


Схема прохождения пакетов из локальной сети к серверу.

Маршрутизаторы объединяют отдельные сети в общую составную сеть (см. рисунок ниже). К каждому маршрутизатору могут быть присоединены несколько сетей (по крайней мере две).

**Маршрут** - это последовательность маршрутизаторов, которые должен пройти пакет от отправителя до пункта назначения.

**1.1.2 Схема объединения отдельных сетей в общую составную сеть.**

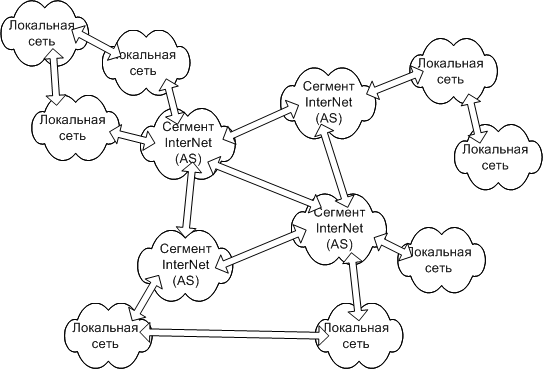


Схема объединения отдельных сетей в общую составную сеть

Локальных сетей слишком много, поэтому реально объединяют автономные системы.

**Автономная система** (AS - autonomous system) - сеть находящаяся под одним административным контролем, это может быть несколько компьютеров или большая сеть (понятие достаточно условное)

**1.1.3 Официальная документация по Internet**

**RFC (Request for Comments)** - официальная документация по Internet, можно найти по адресу <http://www.rfc-editor.org/> или <http://www.ietf.org/rfc.html> .

Все разработчики должны придерживаться этой документации, но на практике, не всегда так происходит.

**1.2 Адресация в сети Internet.**

**1.2.1 Типы адресов.**

Типы адресов:

1. **Физический (MAC-адрес)**
2. **Сетевой (IP-адрес)**
3. **Символьный (DNS-имя)**

Компьютер в сети TCP/IP может имееть адреса трех уровней (но не менее двух):

* Локальный адрес компьютера. Для узлов, входящих в локальные сети - это МАС-адрес сетевого адаптера. Эти адреса назначаются производителями оборудования и являются уникальными адресами.
* IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов.
* Символьный идентификатор-имя (DNS), например, [www.kstu.ru](http://www.kstu.ru).

**1.2.2 IP-адреса**

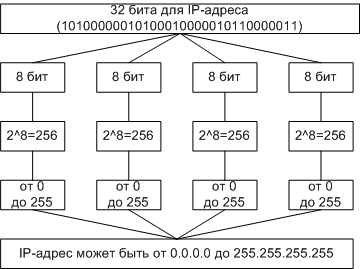
**IPv4** - адрес является уникальным 32-битным идентификатором IP-интерфейса в Интернет.

**IPv6** - адрес является уникальным 128-битным идентификатором IP-интерфейса в Интернет, иногда называют **Internet-2,** адресного пространства IPv4 уже стало не хватать, поэтому постепенно вводят новый стандарт.

IP-адреса принято записывать разбивкой всего адреса по октетам (8), каждый октет записывается в виде десятичного числа, числа разделяются точками. Например, адрес

10100000010100010000010110000011   
записывается как

10100000.01010001.00000101.10000011 = 160.81.5.131



Перевод адреса из двоичной системы в десятичную  
  
IP-адрес хоста состоит из номера IP-сети, который занимает старшую область адреса, и номера хоста в этой сети, который занимает младшую часть.

160.81.5.131 - IP-адрес

160.81.5. - номер сети

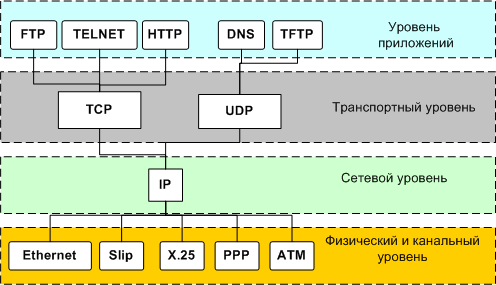
131 -  номер хоста

**1.3 Базовые протоколы (IP, TCP, UDP)**

**1.3.1 Стек протоколов TCP/IP**

TCP/IP - собирательное название для набора (стека) сетевых протоколов разных уровней, используемых в Интернет. Особенности TCP/IP:

* Открытые стандарты протоколов, разрабатываемые независимо от программного и аппаратного обеспечения;
* Независимость от физической среды передачи;
* Система уникальной адресации;
* Стандартизованные протоколы высокого уровня для распространенных пользовательских сервисов.



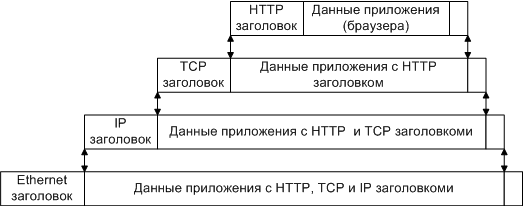
Стек протоколов TCP/IP

Стек протоколов TCP/IP делится на 4 уровня:

* Прикладной,
* Транспортный,
* Межсетевой,
* Физический и канальный.

Позже была принята 7-ми уровневая модель ISO, но она не используется.

Данные передаются в пакетах. Пакеты имеют заголовок и окончание, которые содержат служебную информацию. Данные,  более верхних уровней вставляются, в пакеты нижних уровней.



Пример инкапсуляции пакетов в стеке TCP/IP

**1.3.2 Физический и канальный уровень.**

Стек TCP/IP не подразумевает использования каких-либо определенных протоколов уровня доступа к среде передачи и физических сред передачи данных. От уровня доступа к среде передачи требуется наличие интерфейса с модулем IP, обеспечивающего передачу IP-пакетов. Также требуется обеспечить преобразование IP-адреса узла сети, на который передается IP-пакет, в MAC-адрес. Часто в качестве уровня доступа к среде передачи могут выступать целые протокольные стеки, тогда говорят об IP поверх ATM, IP поверх IPX, IP поверх X.25 и т.п.

**1.3.3 Межсетевой уровень и протокол IP.**

Основу этого уровня составляет IP-протокол.

**IP (Internet Protocol)** – интернет протокол.

Первый стандарт IPv4 определен в RFC-760 (DoD standard Internet Protocol J. Postel Jan-01-1980)

Последняя версия IPv4 - [RFC-791](http://www.ipm.kstu.ru/it/doc/rfc/rfc791.txt) (Internet Protocol J. Postel Sep-01-1981).

Первый стандарт IPv6 определен в RFC-1883 (Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden December 1995)

Последняя версия IPv6 - [RFC-2460](http://www.ipm.kstu.ru/it/doc/rfc/rfc2460.txt) (Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden December 1998).

Основные задачи:

* Адресация
* Маршрутизация
* Фрагментация датаграмм
* Передача данных

Протокол IP доставляет блоки данных от одного IP-адреса к другому.

*Программа, реализующая функции того или иного протокола, часто называется модулем, например, “IP-модуль”, “модуль TCP”.*

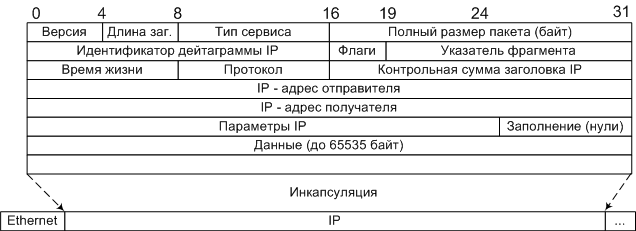
Когда модуль IP получает IP-пакет с нижнего уровня, он проверяет IP-адрес назначения.

* Если IP-пакет адресован данному компьютеру, то данные из него передаются на обработку модулю вышестоящего уровня (какому конкретно - указано в заголовке IP-пакета).
* Если же адрес назначения IP-пакета - чужой, то модуль IP может принять два решения: первое - уничтожить IP-пакет, второе - отправить его дальше к месту назначения, определив маршрут следования - так поступают маршрутизаторы.

Также может потребоваться, на границе сетей с различными характеристиками, разбить IP-пакет на фрагменты

(**фрагментация**), а потом собрать в единое целое на компьютере-получателе.

Если модуль IP по какой-либо причине не может доставить IP-пакет, он уничтожается. При этом модуль IP может отправить компьютеру-источнику этого IP-пакета уведомление об ошибке; такие уведомления отправляются с помощью протокола **ICMP**, являющегося неотъемлемой частью модуля IP. Более никаких средств контроля корректности данных, подтверждения их доставки, обеспечения правильного порядка следования IP-пакетов, предварительного установления соединения между компьютерами протокол IP не имеет. Эта задача возложена на транспортный уровень.



Структура дейтограммы IP. Слова по 32 бита.

**Версия -** версия протокола IP (например, 4 или 6)

**Длина заг.** - длина заголовка IP-пакета.

**Тип сервиса** (TOS - type of service) - Тип сервиса (подробнее рассмотрен в лекции 8).

TOS играет важную роль в маршрутизации пакетов. Интернет не гарантирует запрашиваемый TOS, но многие маршрутизаторы учитывают эти запросы при выборе маршрута (протоколы OSPF и IGRP).

**Идентификатор дейтаграммы, флаги (3 бита)  и указатель фрагмента** - используются для распознавания пакетов, образовавшихся путем  фрагментации исходного пакета.

**Время жизни (TTL - time to live)** - каждый маршрутизатор уменьшает его на 1, что бы пакеты не блуждали вечно.

**Протокол** - Идентификатор протокола верхнего уровня указывает, какому протоколу верхнего уровня принадлежит пакет (например: TCP, UDP).

Коды некоторые протоколов  [RFC-1700](http://www.ipm.kstu.ru/it/doc/rfc/rfc1700.txt)  (1994)

|  |  |  |
| --- | --- | --- |
| **Код** | **Протокол** | **Описание** |
| 0 | - | Зарезервировано |
| 1 | ICMP | Протокол контрольных сообщений |
| 2 | IGMP | Групповой протокол управления |
| 4 | IP | IP-поверх-IP (туннели) |
| 6 | TCP | Протокол управления передачей |
| 8 | EGP | Протокол внешней маршрутизации |
| 9 | IGP | Протокол внутренней маршрутизации |
| 17 | UDP | Протокол дейтограмм пользователя |
| 35 | IDRP | Междоменный протокол маршрутизации |
| 36 | XTP | Xpress транспортный протокол |
| 46 | RSVP | Протокол резервирования ресурсов канала |
| 88 | IGRP | внутренний протокол маршрутизации |
| 89 | OSPFIGP | внутренний протокол маршрутизации |
| 97 | ETHERIP | Ethernet-поверх-IP |
| 101-254 | - | не определены |
| 255 | - | зарезервировано |

1.3.3.1 Маршрутизация.

Протокол IP является маршрутизируемый, для его маршрутизации нужна маршрутная информация.

Маршрутная информация, может быть:

* Статической (маршрутные таблицы прописываются вручную)
* Динамической (маршрутную информацию распространяют специальные протоколы)

Протоколы динамической маршрутизации:

* RIP (Routing Information Protocol) - протокол передачи маршрутной информации, маршрутизаторы динамически создают маршрутные таблицы.
* OSPF (Open Shortest Path First) - протокол "Открой кротчайший путь первым", является внутренним протоколом маршрутизации.
* IGP (Interior Gateway Protocols) - внутренние протоколы маршрутизации, распространяет маршрутную информацию внутри одной автономной системе.
* EGP (Exterior Gateway Protocols) - внешние протоколы маршрутизации, распространяет маршрутную информацию между автономными системами.
* BGP (Border Gateway Protocol) - протокол граничных маршрутизаторов.

1.3.3.2 Протокол ICMP

**ICMP (Internet Control Message Protocol)** - расширение протокола IP, позволяет передавать сообщения об ошибке или проверочные сообщения.

1.3.3.3 Другие служебные IP-протоколы

IGMP (Internet Group Management Protocol) - позволяет организовать многоадресную рассылку средствами IP.

RSVP (Resource Reservation Protocol) - протокол резервирования ресурсов.

ARP (Address Resolution Protocol) - протокол преобразования IP-адреса и адреса канального уровня.

**1.3.4 Транспортный уровень**

Протоколы транспортного уровня обеспечивают прозрачную доставку данных между двумя прикладными процессами. Процесс, получающий или отправляющий данные с помощью транспортного уровня, идентифицируется на этом уровне номером, который называется номером порта. Таким образом, роль адреса отправителя и получателя на транспортном уровне выполняет номер порта (или проще - порт).

Анализируя заголовок своего пакета, полученного от межсетевого уровня, транспортный модуль определяет по номеру порта получателя, какому из прикладных процессов направлены данные, и передает эти данные соответствующему прикладному процессу. Номера портов получателя и отправителя записываются в заголовок транспортным модулем, отправляющим данные; заголовок транспортного уровня содержит также и другую служебную информацию; формат заголовка зависит от используемого транспортного протокола.

На транспортном уровне работают два основных протокола: UDP и TCP.

1.3.4.1 Протокол надежной доставки сообщений TCP

**TCP (Transfer Control Protocol)** – протокол контроля передачи, протокол TCP применяется в тех случаях, когда требуется гарантированная доставка сообщений.

Первая и последняя версия TCP - [RFC-793](http://www.ipm.kstu.ru/it/doc/rfc/rfc793.txt) (Transmission Control Protocol J. Postel Sep-01-1981).

Основные особенности:

* Устанавливается соединение.
* Данные передаются **сегментами**. Модуль TCP нарезает большие сообщения (файлы) на пакеты, каждый из которых передается отдельно, на приемнике наоборот файлы собираются. Для этого нужен **порядковый номер (Sequence Number - SN)** пакета.
* Посылает запрос на следующий пакет, указывая его номер в поле **"Номер подтверждения" (AS).** Тем самым, подтверждая получение предыдущего пакета.
* Делает проверку целостности данных, если пакет битый посылает повторный запрос.



Структура дейтограммы TCP. Слова по 32 бита.

**Длина заголовка -** задается словами по 32бита.

**Размер окна** - количество байт, которые готов принять получатель без подтверждения.

**Контрольная сумма** - включает псевдо заголовок, заголовок и данные.

**Указатель срочности** - указывает последний байт срочных данных, на которые надо немедленно реагировать.

**URG -** флаг срочности, включает поле "Указатель срочности", если =0 то поле игнорируется.

**ACK -** флаг подтверждение, включает поле "Номер подтверждения, если =0 то поле игнорируется.

**PSH -** флаг требует выполнения операции push, модуль TCP должен срочно передать пакет программе.

**RST -** флаг прерывания соединения, используется для отказа в соединении

**SYN -** флаг синхронизация порядковых номеров, используется при установлении соединения.

**FIN -** флаг окончание передачи со стороны отправителя

1.3.4.2 Протокол UDP

**UDP (Universal Datagram Protocol)** **-** универсальный протокол передачи данных, более облегченный транспортный протокол, чем TCP.

Первая и последняя версия UDP - [RFC-768](http://www.ipm.kstu.ru/it/doc/rfc/rfc768.txt) (User Datagram Protocol J. Postel Aug-28-1980).

Основные отличия от TCP:

* Отсутствует соединение между модулями UDP.
* Не разбивает сообщение для передачи
* При потере пакета запрос для повторной передачи не посылается

UDP используется если не требуется гарантированная доставка пакетов , например, для потокового видео и аудио, DNS (т.к. данные небольших размеров). Если проверка контрольной суммы выявила ошибку или если процесса, подключенного к требуемому порту, не существует, пакет игнорируется (уничтожается). Если пакеты поступают быстрее, чем модуль UDP успевает их обрабатывать, то поступающие пакеты также игнорируются.



Структура дейтограммы UDP. Слова по 32 бита.

Не все поля UDP-пакета обязательно должны быть заполнены. Если посылаемая дейтаграмма не предполагает ответа, то на месте адреса отправителя могут помещаться нули.

1.3.4.2 Протокол реального времени RTP

**RTP (Real Time Protocol)** - транспортный протокол для приложений реального времени.

**RTCP (Real Time Control Protocol)** - транспортный протокол обратной связи для приложения RTP..

**1.4 Назначение портов**

По номеру порта транспортные протоколы определяют, какому приложению передать содержимое пакетов.

Порты могут принимать значение от 0-65535 (два байта 2^16).

Номера портам присваиваются таким образом: имеются стандартные номера (например, номер 21 закреплен за сервисом FTP, 23 - за telnet, 80 - за HTTP), а менее известные приложения пользуются произвольно выбранными локальными номерами (как правило, больше>1024), некоторые из них также зарезервированы.

Некоторые заданные порты  [RFC-1700](http://www.ipm.kstu.ru/it/doc/rfc/rfc1700.txt)  (1994)

|  |  |  |
| --- | --- | --- |
| **Порт** | **Служба** | **Описание** |
| 0 | - | Зарезервировано |
| 13 | Daytime | Синхронизация времени |
| 20 | ftp-data | Канал передачи данных для FTP |
| 21 | ftp | Передача файлов |
| 23 | telnet | Сетевой терминал |
| 25 | SMTP | Передача почты |
| 37 | time | Синхронизация времени |
| 43 | Whois | Служба Whois |
| 53 | DNS | Доменные имена |
| 67 | bootps | BOOTP и DHCP - сервер |
| 68 | bootps | BOOTP и DHCP - клиент |
| 69 | tftp | Упрощенная передача почты |
| 80 | HTTP | Передача гипертекста |
| 109 | POP2 | Получение почты |
| 110 | POP3 | Получение почты |
| 119 | NNTP | Конференции |
| 123 | NTP | Синхронизация времени |
| 137 | netbios-ns | NETBIOS - имена |
| 138 | netbios-dgm | NETBIOS Datagram Service |
| 139 | netbios-ssn | NETBIOS Session Service |
| 143 | imap2 | Получение почты |
| 161 | SNMP | Протокол управления |
| 210 | z39.50 | Библиотечный протокол |
| 213 | IPX | IPX - протокол |
| 220 | imap3 | Получение почты |
| 443 | HTTPs | HTTP  с шифрованием |
| 520 | RIP | Динамическая маршрутизация |
| **Диапазон 1024-65535** | | |
| 1024 | - | Зарезервировано |
| 6000-6063 | X11 | Графический сетевой терминал |

Материал из Википедии — свободной энциклопедии

|  |  |
| --- | --- |
| **IP** | |
| **Название:** | Internet protocol |
| **Уровень (по** [**модели OSI**](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI)**):** | Сетевой |
| **Семейство:** | [TCP/IP](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%B2_TCP/IP) |
| **Создан в:** | [1981](http://ru.wikipedia.org/wiki/1981) г. |
| **Назначение протокола:** | Передача [датаграмм](http://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) между хостами сетей [TCP/IP](http://ru.wikipedia.org/wiki/TCP/IP) |
| **Спецификация:** | [RFC 791](http://tools.ietf.org/html/rfc791) |

*У этого термина существуют и другие значения, см.* [*IP (значения)*](http://ru.wikipedia.org/wiki/IP_%28%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%29)*.*

[англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ***Internet Protocol*** — межсетевой [протокол](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB). Относится к [маршрутизируемым](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) протоколам [сетевого уровня](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) семейства [TCP/IP](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%B2_TCP/IP).

Протокол IP используется для негарантированной доставки данных, разделяемых на так называемые [пакеты](http://ru.wikipedia.org/wiki/IP-%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) от одного [узла сети](http://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B5%D0%BB_%D1%81%D0%B5%D1%82%D0%B8) к другому. Это означает, что на уровне этого протокола (третий уровень [сетевой модели OSI](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI)) не даётся гарантий надёжной доставки пакета до адресата. В частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (когда приходят две копии одного пакета; в реальности это бывает крайне редко), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прибыть вовсе. Гарантию безошибочной доставки пакетов дают протоколы более высокого ([транспортного уровня](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)) [сетевой модели OSI](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI) — например, [TCP](http://ru.wikipedia.org/wiki/TCP) — которые используют IP в качестве транспорта.

|  |
| --- |
| **Содержание**   [[убрать](http://ru.wikipedia.org/wiki/IP)]   * [1 Версия 4](http://ru.wikipedia.org/wiki/IP#.D0.92.D0.B5.D1.80.D1.81.D0.B8.D1.8F_4) * [2 Версия 6](http://ru.wikipedia.org/wiki/IP#.D0.92.D0.B5.D1.80.D1.81.D0.B8.D1.8F_6) * [3 Пакет (датаграмма)](http://ru.wikipedia.org/wiki/IP#.D0.9F.D0.B0.D0.BA.D0.B5.D1.82_.28.D0.B4.D0.B0.D1.82.D0.B0.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.B0.29)   + [3.1 Версия 4 (IPv4)](http://ru.wikipedia.org/wiki/IP#.D0.92.D0.B5.D1.80.D1.81.D0.B8.D1.8F_4_.28IPv4.29)   + [3.2 Версия 6 (IPv6)](http://ru.wikipedia.org/wiki/IP#.D0.92.D0.B5.D1.80.D1.81.D0.B8.D1.8F_6_.28IPv6.29) * [4 См. также](http://ru.wikipedia.org/wiki/IP#.D0.A1.D0.BC._.D1.82.D0.B0.D0.BA.D0.B6.D0.B5) * [5 Ссылки](http://ru.wikipedia.org/wiki/IP#.D0.A1.D1.81.D1.8B.D0.BB.D0.BA.D0.B8) * [6 Примечания](http://ru.wikipedia.org/wiki/IP#.D0.9F.D1.80.D0.B8.D0.BC.D0.B5.D1.87.D0.B0.D0.BD.D0.B8.D1.8F) |

**[**[**править**](http://ru.wikipedia.org/w/index.php?title=IP&action=edit&section=1)**] Версия 4**

*Основная статья*: [***IPv4***](http://ru.wikipedia.org/wiki/IPv4)

В современной сети [Интернет](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) используется IP четвёртой версии, также известный как IPv4. В протоколе IP этой версии каждому узлу сети ставится в соответствие [IP-адрес](http://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81) длиной 4 [октета](http://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D0%B5%D1%82_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) (4 [байта](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82)). При этом компьютеры в [подсетях](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F) объединяются общими начальными [битами](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) [адреса](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81). Количество этих бит, общее для данной подсети, называется [маской подсети](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%BA%D0%B0_%D0%BF%D0%BE%D0%B4%D1%81%D0%B5%D1%82%D0%B8) (ранее использовалось деление пространства адресов по [классам](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F) — A, B, C; класс сети определялся диапазоном значений старшего октета и определял число адресуемых узлов в данной сети, сейчас используется [бесклассовая адресация](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F)).

**[**[**править**](http://ru.wikipedia.org/w/index.php?title=IP&action=edit&section=2)**] Версия 6**

*Основная статья*: [***IPv6***](http://ru.wikipedia.org/wiki/IPv6)

В настоящее время вводится в эксплуатацию шестая версия протокола — [IPv6](http://ru.wikipedia.org/wiki/IPv6), которая позволяет адресовать значительно большее количество узлов, чем IPv4. Эта версия отличается повышенной разрядностью адреса, встроенной возможностью шифрования и некоторыми другими особенностями. Переход с IPv4 на IPv6 связан с трудоёмкой работой операторов связи и производителей программного обеспечения и не может быть выполнен одномоментно. На середину [2010 года](http://ru.wikipedia.org/wiki/2010_%D0%B3%D0%BE%D0%B4) в [Интернете](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) присутствовало более 3000 сетей, работающих по протоколу IPv6. Для сравнения, на то же время в адресном пространстве IPv4 присутствовало более 320 тысяч сетей, но в IPv6 сети гораздо более крупные, нежели в IPv4.

**[**[**править**](http://ru.wikipedia.org/w/index.php?title=IP&action=edit&section=3)**] Пакет (датаграмма)**

**IP-пакет** — форматированный блок [информации](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F), передаваемый по [вычислительной сети](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Соединения вычислительных сетей, которые не поддерживают пакеты, такие как традиционные соединения типа «точка-точка» в телекоммуникациях, просто передают данные в виде последовательности [байтов](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82), [символов](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB) или [битов](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82). При использовании пакетного форматирования сеть может передавать длинные сообщения более надежно и эффективно.

**[**[**править**](http://ru.wikipedia.org/w/index.php?title=IP&action=edit&section=4)**] Версия 4 (IPv4)**

*Основная статья*: [***IPv4***](http://ru.wikipedia.org/wiki/IPv4)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Версия** | | | | **IHL** | | | | [**Тип обслуживания**](http://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%BE%D0%B1%D1%81%D0%BB%D1%83%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) | | | | | | | | **Длина пакета** | | | | | | | | | | | | | | | |
| **Идентификатор** | | | | | | | | | | | | | | | | **Флаги** | | | **Смещение фрагмента** | | | | | | | | | | | | |
| **Время жизни (**[**TTL**](http://ru.wikipedia.org/wiki/Time_to_live)**)** | | | | | | | | **Протокол** | | | | | | | | **Контрольная сумма заголовка** | | | | | | | | | | | | | | | |
| **IP-адрес отправителя (32 бита)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **IP-адрес получателя (32 бита)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Параметры (от 0 до 10-ти 32-х битных слов)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Данные (до 65535 байт минус заголовок)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

* Версия — для IPv4 значение поля должно быть равно 4.
* IHL — длина заголовка IP-пакета в 32-битных словах (dword). Именно это поле указывает на начало блока данных в пакете. Минимальное корректное значение для этого поля равно 5.
* Идентификатор — значение, назначаемое отправителем пакета и предназначенное для определения корректной последовательности фрагментов при сборке датаграммы. Для фрагментированного пакета все фрагменты имеют одинаковый идентификатор.
* 3 бита флагов. Первый бит должен быть всегда равен нулю, второй бит DF (don’t fragment) определяет возможность фрагментации пакета и третий бит MF (more fragments) показывает, не является ли этот пакет последним в цепочке пакетов.
* Смещение фрагмента — значение, определяющее позицию фрагмента в потоке данных.
* Время жизни ([TTL](http://ru.wikipedia.org/wiki/Time_to_live)) — число маршрутизаторов, которые должен пройти этот пакет. При прохождении маршрутизатора это число уменьшатся на единицу. Если значения этого поля равно нулю то, пакет должен быть отброшен и отправителю пакета может быть послано сообщение *Time Exceeded* ([ICMP](http://ru.wikipedia.org/wiki/ICMP) код 11 тип 0).
* Протокол — идентификатор интернет-протокола следующего уровня указывает, данные какого протокола содержит пакет, например, TCP или ICMP (см. [IANA protocol numbers](http://www.iana.org/assignments/protocol-numbers) и [RFC 1700](http://tools.ietf.org/html/rfc1700)). В [IPv6](http://ru.wikipedia.org/wiki/IPv6) называется «Next Header».
* Контрольная сумма заголовка — вычисляется с использованием операций поразрядного сложения 16-разрядных слов заголовка по модулю 2. Сама контрольная сумма является дополнением по модулю один полученного результата сложения.

**[**[**править**](http://ru.wikipedia.org/w/index.php?title=IP&action=edit&section=5)**] Версия 6 (**[**IPv6**](http://ru.wikipedia.org/wiki/IPv6)**)**

*Основная статья*: [***IPv6***](http://ru.wikipedia.org/wiki/IPv6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Позиция в октетах** |  | **0** | | | | | | | | **1** | | | | | | | | **2** | | | | | | | | **3** | | | | | | | |
|  | **Позиция в битах** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** |
| **0** | **0** | Версия | | | | [Класс трафика](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%D0%BE%D0%B1%D1%81%D0%BB%D1%83%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) | | | | | | | | [Метка потока](http://ru.wikipedia.org/wiki/IPv6#.D0.9C.D0.B5.D1.82.D0.BA.D0.B8_.D0.BF.D0.BE.D1.82.D0.BE.D0.BA.D0.BE.D0.B2) | | | | | | | | | | | | | | | | | | | |
| **4** | **32** | Длина полезной нагрузки | | | | | | | | | | | | | | | | След. заголовок | | | | | | | | Число переходов | | | | | | | |
| **8** | **64** | IP-адрес отправителя | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **12** | **96** |
| **16** | **128** |
| **20** | **160** |
| **24** | **192** | IP-адрес получателя | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **28** | **224** |
| **32** | **256** |
| **36** | **288** |

* Версия — для IPv6 значение поля должно быть равно 6.
* Класс трафика — определяет приоритет трафика ([QoS](http://ru.wikipedia.org/wiki/QoS), [класс обслуживания](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%D0%BE%D0%B1%D1%81%D0%BB%D1%83%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).
* Метка потока — уникальное число, одинаковое для однородного потока пакетов.
* Длина полезной нагрузки — длина данных (заголовок IP-пакета не учитывается).
* Следующий заголовок — задаёт тип расширенного заголовка ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *IPv6 extension*), который идёт следующим. В последнем расширенном заголовке поле *Next header* задаёт тип транспортного протокола ([TCP](http://ru.wikipedia.org/wiki/TCP), [UDP](http://ru.wikipedia.org/wiki/UDP) и т. д.) и определяет следующий [инкапсулированный](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8%29) уровень.
* Число переходов — максимальное число [маршрутизаторов](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80), которые может пройти пакет. При прохождении маршрутизатора это значение уменьшается на единицу и по достижении нуля пакет отбрасывается.
* Основные [протоколы](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [TCP/IP](http://ru.wikipedia.org/wiki/TCP/IP) по уровням [модели OSI](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI)
* [Прикладной](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) [BGP](http://ru.wikipedia.org/wiki/Border_Gateway_Protocol) • [HTTP](http://ru.wikipedia.org/wiki/HTTP) • [HTTPS](http://ru.wikipedia.org/wiki/HTTPS) • [DHCP](http://ru.wikipedia.org/wiki/DHCP) • [IRC](http://ru.wikipedia.org/wiki/IRC) • [SNMP](http://ru.wikipedia.org/wiki/SNMP) • [DNS](http://ru.wikipedia.org/wiki/DNS) • [DNSSEC](http://ru.wikipedia.org/wiki/DNSSEC) • [NNTP](http://ru.wikipedia.org/wiki/NNTP) • [XMPP](http://ru.wikipedia.org/wiki/XMPP) • [SIP](http://ru.wikipedia.org/wiki/SIP) • [BitTorrent](http://ru.wikipedia.org/wiki/BitTorrent) • [IPP](http://ru.wikipedia.org/wiki/Internet_Printing_Protocol) • [NTP](http://ru.wikipedia.org/wiki/NTP) • [SNTP](http://ru.wikipedia.org/wiki/SNTP)

|  |  |
| --- | --- |
| [***Электронная почта***](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) | [SMTP](http://ru.wikipedia.org/wiki/SMTP)  [POP3](http://ru.wikipedia.org/wiki/POP3)  [IMAP](http://ru.wikipedia.org/wiki/IMAP)4 |
| ***Передача файлов*** | [FTP](http://ru.wikipedia.org/wiki/FTP) • [TFTP](http://ru.wikipedia.org/wiki/TFTP) • [SFTP](http://ru.wikipedia.org/wiki/SFTP) |
| ***Удалённый доступ*** | [rlogin](http://ru.wikipedia.org/wiki/Rlogin) • [Telnet](http://ru.wikipedia.org/wiki/Telnet) • [RDP](http://ru.wikipedia.org/wiki/Remote_Desktop_Protocol) |

* [Представления](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) [XDR](http://ru.wikipedia.org/wiki/External_Data_Representation) • [SSL](http://ru.wikipedia.org/wiki/SSL)
* [Сеансовый](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) [ADSP](http://ru.wikipedia.org/wiki/AppleTalk) • [H.245](http://ru.wikipedia.org/wiki/H.245) • [iSNS](http://ru.wikipedia.org/wiki/Internet_Storage_Name_Service) • [NetBIOS](http://ru.wikipedia.org/wiki/NetBIOS) • [PAP](http://ru.wikipedia.org/wiki/Password_Authentication_Protocol) • [RPC](http://ru.wikipedia.org/wiki/Remote_Procedure_Call) • [L2TP](http://ru.wikipedia.org/wiki/L2TP) • [PPTP](http://ru.wikipedia.org/wiki/PPTP) • [RTCP](http://ru.wikipedia.org/wiki/RTCP) • [SMPP](http://ru.wikipedia.org/wiki/SMPP) • [SCP](http://ru.wikipedia.org/wiki/SCP) • [SSH](http://ru.wikipedia.org/wiki/SSH) • [ZIP](http://ru.wikipedia.org/wiki/Zone_Information_Protocol) • [SDP](http://ru.wikipedia.org/wiki/Session_Description_Protocol)
* [Транспортный](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) [TCP](http://ru.wikipedia.org/wiki/TCP) • [UDP](http://ru.wikipedia.org/wiki/UDP) • [SCTP](http://ru.wikipedia.org/wiki/SCTP) • [DCCP](http://ru.wikipedia.org/wiki/DCCP) • [RUDP](http://ru.wikipedia.org/wiki/RUDP) • [RTP](http://ru.wikipedia.org/wiki/RTP)
* [Сетевой](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) [IPv4](http://ru.wikipedia.org/wiki/IPv4) • [IPv6](http://ru.wikipedia.org/wiki/IPv6) • [IPsec](http://ru.wikipedia.org/wiki/IPsec) • [ICMP](http://ru.wikipedia.org/wiki/ICMP) • [IGMP](http://ru.wikipedia.org/wiki/IGMP) • [ARP](http://ru.wikipedia.org/wiki/ARP) • [RARP](http://ru.wikipedia.org/wiki/RARP) • [RIP2](http://ru.wikipedia.org/wiki/RIP2) • [OSPF](http://ru.wikipedia.org/wiki/OSPF)
* [Канальный](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) [Ethernet](http://ru.wikipedia.org/wiki/Ethernet) • [PPPoE](http://ru.wikipedia.org/wiki/PPPoE) • [PPP](http://ru.wikipedia.org/wiki/PPP_%28%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%29) • [L2F](http://ru.wikipedia.org/wiki/Layer_2_Forwarding_Protocol) • [802.11 Wi-Fi](http://ru.wikipedia.org/wiki/IEEE_802.11) • [802.16 WiMax](http://ru.wikipedia.org/wiki/Wireless_MAN) • [Token ring](http://ru.wikipedia.org/wiki/Token_ring) • [ARCNET](http://ru.wikipedia.org/wiki/ARCNET) • [FDDI](http://ru.wikipedia.org/wiki/FDDI) • [HDLC](http://ru.wikipedia.org/wiki/HDLC) • [SLIP](http://ru.wikipedia.org/wiki/SLIP) • [ATM](http://ru.wikipedia.org/wiki/ATM) • [DTM](http://ru.wikipedia.org/wiki/DTM_%28%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%29) • [X.25](http://ru.wikipedia.org/wiki/X.25) • [Frame relay](http://ru.wikipedia.org/wiki/Frame_relay) • [SMDS](http://ru.wikipedia.org/wiki/SMDS) • [STP](http://ru.wikipedia.org/wiki/STP)
* [Физический](http://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) [Ethernet](http://ru.wikipedia.org/wiki/Ethernet) • [RS-232](http://ru.wikipedia.org/wiki/RS-232) • [EIA-422](http://ru.wikipedia.org/wiki/EIA-422) • [RS-449](http://ru.wikipedia.org/wiki/RS-449)

В Интернет используется много различных типов пакетов, но один из основных - IP-пакет (RFC-791), именно он вкладывается в кадр Ethernet и именно в него вкладываются пакеты UDP, TCP. IP-протокол предлагает ненадежную транспортную среду. Ненадежную в том смысле, что не существует гарантии благополучной доставки IP-дейтограммы. Алгоритм доставки в рамках данного протокола предельно прост: при ошибке дейтограмма выбрасывается, а отправителю посылается соответствующее ICMP-сообщение (или не посылается ничего). Обеспечение же надежности возлагается на более высокий уровень (UDP или TCP). Формат IP-пакетов показан на рисунке 4.4.1.1.



Рис. 4.4.1.1. Формат дейтограммы Интернет

Поле *версия* характеризует версию IP-протокола (например, 4 или 6). Формат пакета определяется программой и, вообще говоря, может быть разным для разных значений поля версия. Только размер и положение этого поля незыблемы. Поэтому в случае изменений длины IP-адреса слишком тяжелых последствий это не вызовет. Понятно также, что значение поля версия во избежании непредсказуемых последствий должно контролироваться программой. *HLEN* - длина заголовка, измеряемая в 32-разрядных словах, обычно заголовок содержит 20 октетов (HLEN=5, без опций и заполнителя). **Заголовок для IPv6 имеет размер в два раза больше, чем для IPv4.** Поле *полная длина* определяет полную длину IP-дейтограммы (до 65535 октетов), включая заголовок и данные. Одно-октетное поле *тип сервиса* (TOS - type of service) характеризует то, как должна обрабатываться дейтограмма. Это поле делится на 6 субполей:

http://book.itep.ru/4/44/tos.gif

Субполе *Приоритет* предоставляет возможность присвоить код приоритета каждой дейтограмме. Значения приоритетов приведены в таблице (в настоящее время это поле не используется).

0 Обычный уровень  
1 Приоритетный  
2 Немедленный  
3 Срочный  
4 Экстренный  
5 ceitic/ecp  
6 Межсетевое управление  
7 Сетевое управление

Формат поля TOS определен в документе RFC-1349. Биты C, D, T и R характеризуют пожелание относительно способа доставки дейтограммы. Так D=1 требует минимальной задержки, T=1 - высокую пропускную способность, R=1 - высокую надежность, а C=1 - низкую стоимость. TOS играет важную роль в маршрутизации пакетов. Интернет не гарантирует запрашиваемый TOS, но многие маршрутизаторы учитывают эти запросы при выборе маршрута (протоколы OSPF и IGRP). В таблице 4.4.1.1 приведены рекомендуемые значения TOS.

**ToS в IP-протоколе**

Таблица 4.4.1.1. Значения TOS для разных протоколов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Процедура | Минимал. задержка | Максим. пропускная способность | Максим. надежность | Минимал. стоимость | Код TOS |
| FTP управление, FTP данные |  |  |  |  |  |
| 1 | 0 | 0 | 0 | 0x10 |
| 0 | 1 | 0 | 0 | 0x08 |
| TFTP | 1 | 0 | 0 | 0 | 0x10 |
| DNS, UDP TCP | 1 | 0 | 0 | 0 | 0x00 |
| 0 | 0 | 0 | 0 | 0x10 |
| 0 | 0 | 0 | 0 | 0x00 |
| telnet | 1 | 0 | 0 | 0 | 0x10 |
| ICMP | 0 | 0 | 0 | 0 | 0x00 |
| IGP | 0 | 0 | 1 | 0 | 0x04 |
| SMTP управление SMTP данные |  |  |  |  |  |
| 1 | 0 | 0 | 0 | 0x10 |
| 0 | 1 | 0 | 0 | 0x08 |
| SNMP | 0 | 0 | 1 | 0 | 0x04 |
| NNTP | 0 | 0 | 0 | 1 | 0x02 |

Только один бит из четырех в TOS может принимать значение 1. Значения по умолчанию равны нулю. Большинство из рекомендаций самоочевидны. Так при telnet наибольшую важность имеет время отклика, а для SNMP (управление сетью) - надежность.

**Замещение ToS на DSCP**

До середины 90-х годов поле TOS в большинстве реализаций игнорировалось. Но после начала разработок средств обеспечения качества обслуживания (QoS) внимание к этому возрасло. Появилось предложение замены поля TOS на поле DSCP (Differenciated Services Code Point), которое также имеет 8 бит (см. RFC-2474). Смотри рис. 4.4.1.1a. Биты CU пока не определены. Иногда это поле называется байтом DS (Differentiated Services).

http://book.itep.ru/4/44/dscp.gif

Рис. 4.4.1.1a. Формат поля DSCP.

Биты DS0-DS5 определяют селектор класса. Значения этого кода представлены в таблице ниже. Стандартным значением DSCP по умолчанию является 000000.

|  |  |
| --- | --- |
| **Селектор класса** | **DSCP** |
| Приоритет 1 | 001000 |
| Приоритет 2 | 010000 |
| Приоритет 3 | 011000 |
| Приоритет 4 | 100000 |
| Приоритет 5 | 101000 |
| Приоритет 6 | 110000 |
| Приоритет 7 | 111000 |

На базе DSCP разработана технология "пошагового поведения" PHB (per Hop Behavior). В рамках этой политики определяются коды DSCP внутри классов. Например, для политики немедленной переадресации EF рекомендуемое значение DSCP=101110. Эта политика соответствует наиболее высокому уровню обслуживания.

Маршрут транспортировки IP-дейтограммы нельзя знать заранее, это связано с поэтапным (по-шаговом) принятием решения о пути каждого пакета. Это свойство маршрутизации обусловлено тем, что IP является протоколом передачи данных без установления соединения.

Поля *идентификатор*, *флаги* (3 бита) и *указатель фрагмента* (fragment offset) управляют процессом фрагментации и последующей "сборки" дейтограммы. *Идентификатор* представляет собой уникальный код дейтограммы, позволяющий идентифицировать принадлежность фрагментов и исключить ошибки при "сборке" дейтограмм. Бит 0 поля *флаги* является резервным, бит 1 служит для управления фрагментацией пакетов (0 - фрагментация разрешена; 1 - запрещена), бит 2 определяет, является ли данный фрагмент последним (0 - последний фрагмент; 1 - следует ожидать продолжения). Поле *время жизни* (TTL - time to live) задает время жизни дейтограммы в секундах, т.е. предельно допустимое время пребывания дейтограммы в системе. При каждой обработке дейтограммы, например в маршрутизаторе, это время уменьшается в соответствии со временем пребывания в данном устройстве или согласно протоколу обработки. Если TTL=0, дейтограмма из системы удаляется. Во многих реализациях TTL измеряется в числе шагов, в этом случае каждый маршрутизатор выполняет операцию TTL=TTL-1. TTL помогает предотвратить зацикливание пакетов. Поле *протокол* аналогично полю *тип* в Ethernet-кадре и определяет структуру поля *данные* (см. табл. 4.4.1.2).

|  |
| --- |
| Поле TTL относится к числу переменных полей заголовка. При прохождении через маршрутизатор над содержимым этого поля производится операция **TTL=TTL-1**, при этом должна быть пересчитана контрольная сумма. И, если TTL=0, дейтограммы отбрасывается. |

Поле *контрольная сумма заголовка* вычисляется с использованием операций сложения 16-разрядных слов заголовка по модулю 1. Сама контрольная сумма является дополнением по модулю один полученного результата сложения. Обратите внимание, здесь осуществляется контрольное суммирование заголовка, а не всей дейтограммы. Поле *опции* не обязательно присутствует в каждой дейтограмме. Размер поля *опции* зависит от того, какие опции применены. Если используется несколько опций, они записываются подряд без каких-либо разделителей. Каждая опция содержит один октет кода опции, за которым может следовать октет длины и серия октетов данных. Если место, занятое опциями, не кратно 4 октетам, используется заполнитель. Структура октета кода опции отражена на рис. 4.4.1.2.

**Коды протоколов Интернет**

Таблица 4.4.1.2. Коды протоколов Интернет

|  |  |  |
| --- | --- | --- |
| Код протокола Интернет | Сокращенное название протокола | Описание |
| 0 | - | Зарезервировано |
| 1 | ICMP | Протокол контрольных сообщений [rfc792] |
| 2 | IGMP | Групповой протокол управления [rfc1112] |
| 3 | GGP | Протокол маршрутизатор-маршрутизатор [RFC-823] |
| 4 | IP | IP поверх IP (инкапсуляция/туннели) |
| 5 | ST | Поток [rfc1190] |
| 6 | TCP | Протокол управления передачей [RFC-793] |
| 7 | UCL | UCL |
| 8 | EGP | Протокол внешней маршрутизации [RFC-888] |
| 9 | IGP | Протокол внутренней маршрутизации |
| 10 | BBN-MON | BBN-RCC мониторирование |
| 11 | NVP-II | Сетевой протокол для голосовой связи [RFC-741] |
| 12 | PUP | PUP |
| 13 | ARGUS | argus |
| 14 | Emcon | emcon |
| 15 | Xnet | Перекрестный сетевой отладчик [IEN158] |
| 16 | Chaos | Chaos |
| 17 | UDP | Протокол дейтограмм пользователя [RFC-768] |
| 18 | MUX | Мультиплексирование [IEN90] |
| 19 | DCN-MEAS | DCN измерительные субсистемы |
| 20 | HMP | Протокол мониторирования ЭВМ (host [RFC-869]) |
| 21 | PRM | Мониторирование при передаче пакетов по радио |
| 22 | XNS-IDP | Xerox NS IDP |
| 23 | Trunk-1 | Trunk-1 |
| 24 | Trank-2 | Trunk-2 |
| 25 | Leaf-1 | Leaf-1 |
| 26 | Leaf-2 | Leaf-2 |
| 27 | RDP | Протокол для надежной передачи данных [RFC-908] |
| 28 | IRTP | Надежный TP для Интернет [RFC-938] |
| 29 | ISO-TP4 | ISO транспортный класс 4 [RFC-905] |
| 30 | Netblt | Массовая передача данных [RFC-969] |
| 31 | MFE-NSP | Сетевая служба MFE |
| 32 | Merit-INP | Межузловой протокол Merit |
| 33 | SEP | Последовательный обмен |
| 34 |  | не определен |
| 35 | IDRP | Междоменный протокол маршрутизации |
| 36 | XTP | Xpress транспортный протокол |
| 37 | DDP | Протокол доставки дейтограмм |
| 38 | IDPR-CMTP | IDPR передача управляющих сообщений |
| 39 | TP++ | TP++ транспортный протокол |
| 40 | IL | IL-транспортный протокол |
| 41 | SIP | Простой Интернет-протокол |
| 42 | SDRP | Протокол маршрутных запросов для отправителя |
| 43 | SIP-SR | SIP исходный маршрут |
| 44 | SIP-Frag | SIP-фрагмент |
| 45 | IDRP | Интер-доменный маршрутный протокол |
| 46 | RSVP | Протокол резервирования ресурсов канала |
| 47 | GRE | Общая инкапсуляция маршрутов |
| 49 | BNA | BNA |
| 50 | SIPP-ESP | SIPP ESЗ |
| 52 | I-NLSP | Интегрированная система безопасности сетевого уровня |
| 53 | Swipe | IP с кодированием |
| 54 | NHRP | nbma протокол определения следующего шага |
| 55-60 |  | не определены |
| 61 |  | Любой внутренний протокол ЭВМ |
| 62 | CFTP | CFTP |
| 63 |  | Любая локальная сеть |
| 64 | Sat-Expak | Satnet и Expak |
| 65 | MIT-Subn | Поддержка субсетей MIT |
| 66 | RVD | Удаленный виртуальный диск MIT |
| 67 | IPPC | IPPC |
| 68 |  | Любая распределенная файловая система |
| 69 | Sat-Mon | Мониторирование Satnet |
| 70 |  | не определен |
| 71 | IPCV | Базовая пакетная утилита |
| 75 | PVP | Пакетный видео-протокол |
| 76 | BRsat-Mon | Резервное мониторирование Satnet |
| 78 | Wb-mon | Мониторирование Expak |
| 79 | Wb-expak | Широкополосная версия Expak |
| 80 | ISO-IP | ISO Интернет протокол |
| 88 | IGRP | IGRP (Cisco) - внутренний протокол маршрутизации |
| 89 | OSPFIGP | OSPFIGP - внутренний протокол маршрутизации |
| 92 | MTP | Транспортный протокол мультикастинга |
| 101-254 |  | не определены |
| 255 |  | зарезервировано |

**Опции IP-протокола**

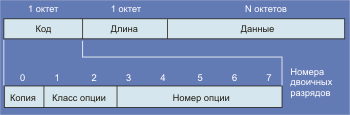


Рис. 4.4.1.2. Формат описания опций

Флаг *копия* равный 1 говорит о том, что опция должна быть скопирована во все фрагменты дейтограммы. При равенстве этого флага 0 опция копируется только в первый фрагмент. Ниже приведены значения разрядов 2-битового поля *класс опции*:

|  |  |
| --- | --- |
| Значение поля *класс опции* | Описание |
| 0 | Дейтограмма пользователя или сетевое управление |
| 1 | Зарезервировано для будущего использования |
| 2 | Отладка и измерения (диагностика) |
| 3 | Зарезервировано для будущего использования |

В таблице, которую вы найдете ниже, приведены значения классов и номеров опций.

|  |  |  |  |
| --- | --- | --- | --- |
| Класс опции | Номер опции | Длина описания | Назначение |
| 0 | 0 | - | Конец списка опций. Используется, если опции не укладываются в поле заголовка (смотри также поле "заполнитель") |
| 0 | 1 | - | Никаких операций (используется для выравнивания октетов в списке опций) |
| 0 | 2 | 11 | Ограничения, связанные с секретностью (для военных приложений) |
| 0 | 3 | \* | Свободная маршрутизация. Используется для того, чтобы направить дейтограмму по заданному маршруту |
| 0 | 7 | \* | Запись маршрута. Используется для трассировки |
| 0 | 8 | 4 | Идентификатор потока. Устарело. |
| 0 | 9 | \* | Жесткая маршрутизация. Используется, чтобы направить дейтограмму по заданному маршруту |
| 2 | 4 | \* | Временная метка Интернет |

\* в колонке "длина" - означает - переменная.

Наибольший интерес представляют собой опции *временные метки* и *маршрутизация*. Опция *записать маршрут* (RR) создает дейтограмму, где зарезервировано место, куда каждый маршрутизатор по дороге должен записать свой IP-адрес (например, утилита traceroute). Формат опции *записать маршрут* в дейтограмме представлен ниже на рис. 4.4.1.3 (предусмотрено место для записи 9 IP-адресов, к сожаления, реализация RR не является обязательной):

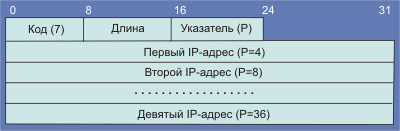


Рис. 4.4.1.3 Формат опций *записать маршрут*

Поле *код* содержит номер опции (7 в данном случае). Поле *длина* определяет размер записи для опций, включая первые 3 октета. *Указатель* отмечает первую свободную позицию в списке IP-адресов (куда можно произвести запись очередного адреса). Интересную возможность представляет опция *маршрут отправителя*, которая открывает возможность посылать дейтограммы по заданному отправителем маршруту. Это позволяет исследовать различные маршруты, в том числе те, которые недоступны через узловые маршрутизаторы. Существует две формы такой маршрутизации: *Свободная маршрутизация* и *Жесткая маршрутизация* (маршрутизация отправителя). Форматы для этих опций показаны ниже:

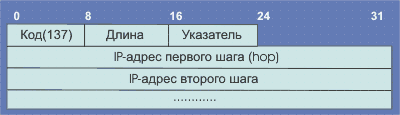


Рис. 4.4.1.3а. Формат опций маршрутизации

*Жесткая маршрутизация* означает, что адреса определяют точный маршрут дейтограммы. Проход от одного адреса к другому может включать только одну сеть. *Свободная маршрутизация* отличается от предшествующей возможностью прохода между двумя адресами списка более чем через одну сеть. Поле *длина* задает размер списка адресов, а *указатель* отмечает адрес очередного маршрутизатора на пути дейтограммы.

IP-слой имеет маршрутные таблицы, которые просматриваются каждый раз, когда IP получает дейтограмму для отправки. Когда дейтограмма получается от сетевого интерфейса, IP первым делом проверяет, принадлежит ли IP-адрес места назначения к списку локальных адресов, или является широковещательным адресом. Если имеет место один из этих вариантов, дейтограмма передается программному модулю в соответствии с кодом в поле протокола. IP-процессор может быть сконфигурирован как маршрутизатор, в этом случае дейтограмма может быть переадресована в другой узел сети. Маршрутизация на IP-уровне носит пошаговый характер. IP не знает всего пути, он владеет лишь информацией - какому маршрутизатору послать дейтограмму с конкретным адресом места назначения.

Просмотр маршрутной таблицы происходит в три этапа:

1. Ищется полное соответствие адресу места назначения. В случае успеха, пакет посылается соответствующему маршрутизатору или непосредственно интерфейсу адресата. Связи точка-точка выявляются именно на этом этапе.
2. Ищется соответствие адресу сети места назначения. В случае успеха система действует также как и в предшествующем пункте. Одна запись в таблице маршрутизации соответствует всем ЭВМ, входящим в данную сеть.
3. Осуществляется поиск маршрута по умолчанию и, если он найден, дейтограмма посылается в соответствующий маршрутизатор.

Для того чтобы посмотреть, как выглядит простая маршрутная таблица, воспользуемся командой netstat -rn (ЭВМ Sun. Флаг -r выводит на экран маршрутную таблицу, а -n отображает IP-адреса в цифровой форме. С целью экономии места таблица в несколько раз сокращена).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| routing tables destination | gateway | flags | refcnt | use | interface |
| 193.124.225.72 | 193.124.224.60 | ughd | 0 | 61 | le0 |
| 192.148.166.1 | 193.124.224.60 | ughd | 0 | 409 | le0 |
| 193.124.226.81 | 193.124.224.37 | ughd | 0 | 464 | le0 |
| 192.160.233.201 | 193.124.224.33 | ughd | 0 | 222 | le0 |
| 192.148.166.234 | 193.124.224.60 | ughd | 1 | 3248 | le0 |
| 193.124.225.66 | 193.124.224.60 | ughd | 0 | 774 | le0 |
| 192.148.166.10 | 193.124.224.60 | ughd | 0 | 621 | le0 |
| 192.148.166.250 | 193.124.224.60 | ughd | 0 | 371 | le0 |
| 192.148.166.4 | 193.124.224.60 | ughd | 0 | 119 | le0 |
| 145.249.16.20 | 193.124.224.60 | ughd | 0 | 130478 | le0 |
| 192.102.229.14 | 193.124.224.33 | ughd | 0 | 13206 | le0 |
| default | 193.124.224.33 | ug | 9 | 5802624 | le0 |
| 193.124.224.32 | 193.124.224.35 | u | 6 | 1920046 | le0 |
| 193.124.134.0 | 193.124.224.50 | ugd | 1 | 291672 | le0 |

Колонка destination - место назначение, Default - отмечает маршрут по умолчанию; Gateway - IP-адреса портов подключения (маршрутизаторов); REFCNT (reference count) - число активных пользователей маршрута; USE - число пакетов, посланных по этому маршруту; interface - условные имена сетевых интерфейсов. Расшифровка поля FLAGS приведено ниже:

|  |  |
| --- | --- |
| u | Маршрут работает (up). |
| g | Путь к маршрутизатору (gateway), если этот флаг отсутствует, адресат доступен непосредственно. |
| h | Маршрут к ЭВМ (host), адрес места назначения является полным адресом этой ЭВМ (адрес сети + адрес ЭВМ). Если флаг отсутствует, маршрут ведет к сети, а адрес места назначения является адресом сети. |
| d | Маршрут возник в результате переадресации. |
| m | Маршрут был модифицирован с помощью переадресации. |

Опция *временные метки* работает также как и опция *запись маршрута*. Каждый маршрутизатор на пути дейтограммы делает запись в одном из полей дейтограммы (два слова по 32 разряда; смотри раздел [4.4.15](http://book.itep.ru/4/44/ntp_4415.htm)). Формат этой опции отображен на рисунке 4.4.1.4.

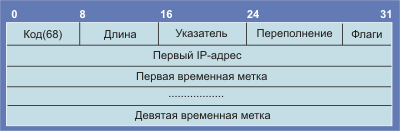


Рис. 4.4.1.4 Формат опции "временные метки"

Смысл полей ***длина*** и ***указатель*** идентичен тому, что сказано о предыдущих опциях. 4-битовое поле ***переполнение*** содержит число маршрутизаторов, которые не смогли записать временные метки из-за ограничений выделенного места в дейтограмме. Значения поля ***флаги*** задают порядок записи временных меток маршрутизаторами:

Таблица 4.4.1.3.

|  |  |
| --- | --- |
| **Значение флага** | **Назначение** |
| 0 | Записать только временные метки; опустить IP-адреса. |
| 1 | Записать перед каждой временной меткой IP-адрес (как в формате на предыдущем рисунке). |
| 3 | IP-адреса задаются отправителем; маршрутизатор записывает только временные метки, если очередной IP-адрес совпадает с адресом маршрутизатора |

Временные метки должны содержать время в миллисекундах, отсчитанное от начала суток. Если маршрутизатору некуда положить свою временную метку (число меток превысило 9), он инкрементирует счетчик ***переполнение***.

Взаимодействие других протоколов с IP можно представить из схемы на рис. 4.4.1.5. В основании лежат протоколы, обеспечивающие обмен информацией на физическом уровне, далее следуют протоколы IP, ICMP, ARP, RARP, IGMP и протоколы маршрутизаторов. Чем выше расположен протокол, тем более высокому уровню он соответствует. Протоколы, имена которых записаны в одной и той же строке, соответствуют одному и тому же уровню. Но все разложить аккуратно по слоям невозможно - некоторые протоколы занимают промежуточное положение, что и отражено на схеме, (области таких протоколов захватывают два уровня. Здесь протоколы IP, ICMP и IGMP помещены на один уровень, для чего имеется не мало причин. Но иногда последние два протокола помещают над IP, так как их пакеты вкладываются в IP-дейтограммы. Так что деление протоколов по уровням довольно условно. На самом верху пирамиды находятся прикладные программы, хотя пользователю доступны и более низкие уровни (например, ICMP), что также отражено на приведенном рисунке (4.4.1.5).

## Распределение протоколов по уровням

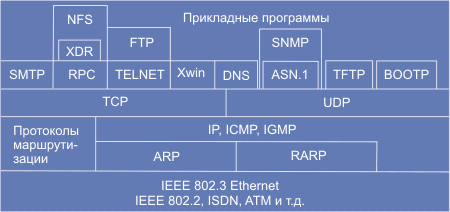


Рис. 4.4.1.5. Распределение протоколов Интернет по уровням

Интернет - это инструмент общения, средство доступа к информации и как всякий инструмент требует практики. Из вашего собственного опыта вы знаете, что можно прочесть ворох инструкций о том, как забивать гвозди, но научиться этому можно лишь на практике. Поэтому рекомендую с самого начала, читая данные тексты, чаще садитесь за терминал.

# 4.4.2 Протокол UDP

#### Семенов Ю.А. (ГНЦ ИТЭФ)

Протокол **UDP** (User Datagram Protocol, RFC-768) является одним из основных протоколов, расположенных непосредственно над IP. Он предоставляет прикладным процессам транспортные услуги, немногим отличающиеся от услуг протокола IP. Протокол UDP обеспечивает доставку дейтограмм, но не требует подтверждения их получения. Протокол UDP не требует соединения с удаленным модулем UDP ("бессвязный" протокол). К заголовку IP-пакета UDP добавляет поля *порт отправителя* и *порт получателя*, которые обеспечивают мультиплексирование информации между различными прикладными процессами, а также поля *длина* UDP-дейтограммы и *контрольная сумма*, позволяющие поддерживать целостность данных. Таким образом, если на уровне IP для определения места доставки пакета используется адрес, на уровне UDP - номер порта.

## Область использования UDP

Примерами сетевых приложений, использующих UDP, являются **NFS** (Network File System), TFTP (Trivial File Transfer protocol, RFC-1350), **RPC** (Remote Procedure Call, RFC-1057) и **SNMP** (Simple Network Management Protocol, RFC-1157). Малые накладные расходы, связанные с форматом UDP, а также отсутствие необходимости подтверждения получения пакета, делают этот протокол наиболее популярным при реализации приложений мультимедиа, но главное его место работы - локальные сети и мультимедиа.

Прикладные процессы и модули UDP взаимодействуют через UDP-порты. Эти порты нумеруются, начиная с нуля. Прикладной процесс, предоставляющий некоторые услуги (сервер), ожидает сообщений, направленных в порт, специально выделенный для этих услуг. Программа-сервер ждет, когда какая-нибудь программа-клиент запросит услугу.

Например, сервер SNMP всегда ожидает сообщения, адресованного в порт 161. Если клиент snmp желает получить услугу, он посылает запрос в UDP-порт 161 на машину, где работает сервер. На каждой машине может быть только один агент SNMP, т.к. существует только один порт 161. Данный номер порта является общеизвестным, т.е. фиксированным номером, официально выделенным в сети Internet для услуг SNMP. Общеизвестные номера портов определяются стандартами Internet (см. табл. 4.4.2.1).

Данные, отправляемые прикладным процессом через модуль UDP, достигают места назначения как единое целое. Например, если процесс-отправитель производит 5 записей в порт, то процесс-получатель должен будет сделать 5 чтений. Размер каждого записанного сообщения будет совпадать с размером каждого прочитанного. Протокол UDP сохраняет границы сообщений, определяемые прикладным процессом. Он никогда не объединяет несколько сообщений в одно и не делит одно сообщение на части. Формат UDP-сообщений представлен ниже на рис. 4.4.2.1:

## Формат UDP-дейтограмм



Рис. 4.4.2.1 Формат UDP-дейтограмм

Длина сообщения равна числу байт в UDP-дейтограмме, включая заголовок. Поле UDP контрольная сумма содержит код, полученный в результате контрольного суммирования UDP-заголовка и поля данные. Не трудно видеть, что этот протокол использует заголовок минимального размера (8 байт). Таблица номеров UDP-портов приведена ниже (4.4.2.1). Номера портов от 0 до 255 стандартизованы и использовать их в прикладных задачах не рекомендуется. Но и в интервале 255-1023 многие номера портов заняты, поэтому прежде чем использовать какой-то порт в своей программе, следует заглянуть в RFC-1700. Во второй колонке содержится стандартное имя, принятое в Internet, а в третей - записаны имена, принятые в UNIX.

**Таблица 4.4.2.1** Номера UDP-портов (более полный перечень в RFC-1700; Если какой-то номер порта в перечне отсутствует, это не означает, что он не зарезервирован и его можно использовать, просто я сэкономил место). См. IANA, а также Приложения.

## Стандартные номера портов UDP

|  |  |  |  |
| --- | --- | --- | --- |
| Десятич. номер порта | Обозначение порта | | Описание |
|  | в Интернет | в Unix |  |
| 0 | - | - | Зарезервировано |
| 1 | TCPmux | - | TCP Мультиплексор |
| 2 | Compressnet | - | Программа управления |
| 3 | Compressnet | - | Процесс сжатия |
| 5 | RJE | - | Вход в удаленную задачу |
| 7 | Echo | echo | Эхо |
| 9 | Discard | discard | Сброс |
| 11 | Users | systat | Активные пользователи |
| 13 | Daytime | daytime | Время дня |
| 15 | - | Netstat | Кто работает или netstat |
| 19 | Chargen | chargen | Генератор символов |
| 20 | FTP-data | ftp-data | FTP (данные) |
| 21 | FTP | ftp | Протокол пересылки файлов (управление) |
| 23 | telnet | telnet | Подключение терминала |
| 24 | - | - | Любая частная почтовая система |
| 25 | SMTP | SMTP | Протокол передачи почтовых сообщений |
| 31 | MSG-auth |  | Распознавание сообщения (аутентификация) |
| 35 | - | - | Любой частный принт-сервер |
| 37 | Time | time | Время |
| 39 | RLP | - | Протокол поиска ресурсов |
| 41 | Graphics |  | Графика |
| 42 | nameserver | name | Сервер имен |
| 43 | Nicname | whois | Кто это? (whois-сервис) |
| 45 | MPM | - | Блок обработки входных сообщений |
| 46 | MPM-snd | - | Блок обработки выходных сообщений |
| 48 | Auditd | - | Демон цифрового аудита |
| 49 | login | - | Протокол входа в ЭВМ |
| 50 | RE-mail-ck | - | Протокол удаленного контроля почтовым обменом |
| 53 | Domain | nameserver | Сервер имен доменов (dns) |
| 57 | - | - | Любой частный терминальный доступ |
| 59 | - | - | Любой частный файл-сервер |
| 64 | covia | - | Коммуникационный интегратор (ci) |
| 66 | SQL\*net | - | Oracle SQL\*net |
| 67 | Bootps | Bootps | Протокол загрузки сервера |
| 68 | Bootpc | bootpc | Протокол загрузки клиента |
| 69 | TFTP | tftp | Упрощенная пересылка файлов |
| 70 | Gopher | - | Gopher (поисковая система) |
| 71 | - | Netrjs-1 | Сервис удаленных услуг |
| 77 | - | rje | Любой частный RJE-сервис |
| 79 | Finger | finger | finger |
| 80 | WWW-HTTP |  | World Wide Web HTTP |
| 81 | Hosts2-NS | - | Сервер имен 2 |
| 87 | - | - | Любая частная терминальная связь |
| 88 | Kerberos |  | Kerberos |
| 92 | NPP | - | Протокол сетевой печати |
| 93 | DCP | - | Протокол управления приборами |
| 95 | Supdup | supdup | Supdup протокол |
| 97 | Swift-rvf | - | swift-протокол удаленных виртуальных файлов |
| 101 | Hostname | hostnames | Сервер имен ЭВМ для сетевого информационного центра |
| 102 | ISO-Tsap | iso-tsap | ISO-Tsap |
| 103 | GPPitnp |  | Сети точка-точка |
| 104 | ACR-nema |  | ACR-nema digital IMAG. & comm. 300 |
| 108 | Snagas |  | sna-сервер доступа |
| 109 | POP2 | - | Почтовый протокол pop2 |
| 110 | POP3 | - | Почтовый протокол POP3 |
| 111 | SUNRPC | sunrpc | SUN microsystem RPC |
| 113 | Auth | auth | Служба распознавания |
| 114 | Audionews |  | Аудио-новости |
| 115 | SFTP |  | Простой протокол передачи файлов |
| 117 | UUCP-path | uucp-path | Служба паролей UUCP |
| 118 | SQLserv |  | SQL-сервер |
| 119 | NNTP | NNTP | Протокол передачи новостей |
| 123 | NTP | NTP | Сетевой протокол синхронизации |
| 129 | PWDgen |  | Протокол генерации паролей |
| 130-132 |  |  | Cisco |
| 133 | Statsrv |  | Сервер статистики |
| 134 | Ingres-net |  | Ingres-net-сервис |
| 135 | LOC-srv |  | Поисковый сервис |
| 137 | Netbios-SSN | - | Служба имен Netbios |
| 138 | Netbios-DGM |  | Служба дейтограмм netbios |
| 139 | Netbios-SSN |  | Служба сессий Netbios |
| 147 | ISO-IP |  | ISO-IP |
| 150 | SQL-net |  | SQL net |
| 152 | BFTP |  | Протокол фоновой пересылки файлов |
| 156 | SQLsrv |  | SQL-сервер |
| 158 | PCmail-srv |  | PC почтовый сервер |
| 161 | - | SNMP | Сетевой монитор SNMP |
| 162 | - | SNMP-trap | SNMP-ловушки |
| 170 | Print-srv |  | postscript сетевой сервер печати |
| 179 | BGP |  | Динамический протокол внешней маршрутизации |
| 191 | Prospero |  | Служба каталогов Prospero |
| 194 | IRC |  | Протокол Интернет для удаленных переговоров |
| 201-206 |  |  | Протоколы сетей Apple talk |
| 213 | IPX |  | ipx |
| 348 | CSI-SGWP |  | Протокол управления cabletron |
| 396 | Netware-IP |  | Novell-Netware через IP |
| 398 | Kryptolan |  | Kryptolan |
| 414 | Infoseek |  | Infoseek (информационный поиск) |
| 418 | Hyper-g |  | Hyper-g |
| 444 | SNPP |  | Простой протокол работы со страницами |
| 512 | - | biff (exec) | Unix Comsat (удаленное исполнение) |
| 513 | - | Who | Unix Rwho daemon |
| 514 | - | syslog | Дневник системы |
| 515 | Printer |  | Работа с буфером печати (spooler) |
| 525 | - | Timed | Драйвер времени |

Зарегистрировано ряд портов для стандартного применения и в диапазоне 1024-65535. Например:

|  |  |  |
| --- | --- | --- |
| **Номер порта** | **Обозначение** | **Назначение** |
| 1397 | Аudio-activmail | Активная звуковая почта |
| 1398 | Video-activmail | Активная видео-почта |
| 5002 | RFE | Радио-Ethernet |
| 6000-6063 | X11 | Система X Window |
| 7008 | AFS3-update | Сервер-сервер актуализация |

## Схема вычисления контрольных сумм

Модуль IP передает поступающий IP-пакет модулю UDP, если в заголовке этого пакета указан код протокола UDP. Когда модуль UDP получает дейтограмму от модуля IP, он проверяет контрольную сумму, содержащуюся в ее заголовке. Если контрольная сумма равна нулю, это означает, что отправитель ее не подсчитал. ICMP, IGMP, UDP и TCP протоколы имеют один и тот же алгоритм вычисления контрольной суммы (RFC-1071). Но вычисление контрольной суммы для UDP имеет некоторые особенности. Во-первых, длина UDP-дейтограммы может содержать нечетное число байт, в этом случае к ней добавляется нулевой байт, который служит лишь для унификации алгоритма и никуда не пересылается. Во-вторых, при расчете контрольной суммы для UDP и TCP добавляются 12-байтные псевдо-заголовки, содержащие IP-адреса отправителя и получателя, код протокола и длину дейтограммы (см. рис. 4.4.2.2). Как и в случае IP-дейтограммы, если вычисленная контрольная сумма равна нулю, в соответствующее поле будет записан код 65535.

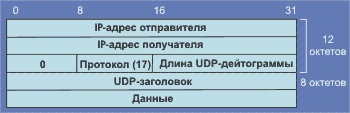


Рис. 4.4.2.2. Псевдозаголовок, используемый при расчете контрольной суммы

Если контрольная сумма правильная (или равна 0), то проверяется порт назначения, указанный в заголовке дейтограммы. Если прикладной процесс подключен к этому порту, то прикладное сообщение, содержащиеся в дейтограмме, становится в очередь к прикладному процессу для прочтения. В остальных случаях дейтограмма отбрасывается. Если дейтограммы поступают быстрее, чем их успевает обрабатывать прикладной процесс, то при переполнении очереди сообщений поступающие дейтограммы отбрасываются модулем UDP. Следует учитывать, что во многих посылках контрольное суммирование не охватывает адреса отправителя и места назначения. При некоторых схемах маршрутизации это приводит к зацикливанию пакетов в случае повреждения его адресной части (адресат не признает его "своим").

Так как максимальная длина IP-дейтограммы равна 65535 байтам, максимальная протяженность информационного поля UDP-дейтограммы составляет 65507 байт. На практике большинство систем работает с UDP-дейтограммами с длиной 8192 байта или менее (Ethernet допускает 1508 байт). Детальное описание форматов, полей пакетов и пр. читатель может найти в RFC-768. Смотри также RFC-2147 (IPv6 Jumbo), RFC-2508 (компрессия заголовков) и RFC-3828 (Lightweight UDP).

Нашел применение UDP и в протоколе Teredo (туннелирование IPv6 для систем NAT).

Протокол TCP (transmission control protocol, RFC-793, -1323, -1644[T/TCP], -2018, -2581, -2582[RENO], -2861, -2873, -2883[SACK], -2923[MTU], -2988[RTO], -3293[GSMP], -3448[TFRC], -3465, -3481) в отличии от UDP осуществляет доставку дейтограмм, называемых сегментами, в виде байтовых потоков с установлением соединения. Протокол TCP применяется в тех случаях, когда требуется гарантированная доставка сообщений. Он использует контрольные суммы пакетов для проверки их целостности и освобождает прикладные процессы от необходимости таймаутов и повторных передач для обеспечения надежности. Для отслеживания подтверждения доставки в TCP реализуется алгоритм "скользящего" окна. Наиболее типичными прикладными процессами, использующими TCP, являются FTP (File Transfer Protocol - протокол передачи файлов) и telnet. Кроме того, TCP используют системы SMTP, HTTP, X-window, RCP (remote copy), а также "r"-команды. Внутренняя структура модуля TCP гораздо сложнее структуры UDP. Подобно UDP прикладные процессы взаимодействуют с модулем TCP через порты (см. [таблицу 4.4.2.1](http://book.itep.ru/4/44/udp_442.htm#point1) в предыдущей главе). Под байтовыми потоками здесь подразумевается то, что один примитив, например, read или write (см. [раздел "Программирование для сетей"](http://book.itep.ru/7/sock_71.htm)) может вызвать посылку адресату последовательности сегментов, которые образуют некоторый блок данных (сообщение). Использование портов открывает возможность осуществлять несколько соединений между двумя сетевыми объектами (работать с разными процессами).

Примером прикладного процесса, использующего TCP, может служить FTP, при этом будет работать стек протоколов ftp/tcp/ip/ethernet. Хотя протоколы UDP и TCP могли бы для сходных задач использовать разные номера портов, обычно этого не происходит. Модули TCP и UDP выполняют функции мультиплексоров/демультиплексоров между прикладными процессами и IP-модулем. При поступлении пакета в модуль IP он будет передан в TCP- или UDP-модуль согласно коду, записанному в поле протокола данного IP-пакета. Формат сегмента (пакета) TCP представлен ниже на рис. 4.4.3.1. Если вы хотите глубже разобраться с особенностями работы этого протокола, рекомендуется воспользоваться услугами программы **tcpdump**, которая позволяет отслеживать содержимое отправляемых и приходящих пакетов в ходе реализации сессии.

Если IP-протокол работает с адресами, то TCP, также как и UDP, с портами. Именно с номеров портов отправителя и получателя начинается заголовок TCP-сегмента. 32-битовое поле ***код позиции в сообщении*** определяет порядковый номер первого октета в поле данных пользователя. В приложениях передатчика и приемника этому полю соответствуют 32-разрядные счетчики числа байт, которые при переполнении обнуляются. При значении флага syn=1 в этом поле лежит код **ISN** (Initial Sequence Number; смотри ниже описание процедуры установления связи), выбираемый для конкретного соединения. Первому байту, передаваемому через созданное соединение, присваивается номер ISN+1. Значение ISN может задаваться случайным образом. Но в UNIX 4.4BSD при загрузке ОС ISN делается равнм 1 (это нарушает требования RFC), а далее увеличивается на 640000 каждые полсекунды. Аналогичная инкрементация осуществляется при установлении нового соединения. В RFC рекомендуется увеличивать счетчик ISN на 1 каждые 4 микросекунды.

32-битовое поле ***номер октета, который должен прийти следующим*** содержит код, который на единицу больше номера номера последнего успешно доставленного (принятого) байта. Содержимое этого поля интерпретируется получателем сегмента, только если присутствует флаг **ACK**. В заголовках всех сегментов, передаваемых после установления соединения это поле заполняется, а флаг AСK=1.

В ТСР предусмотрен режим полнодуплексной передачи. При этом данные могут передаваться в обоих направлениях независимо. В ходе обмена каждая из сторон должна отслеживать позиционные номера передаваемых и принимаемых байт. Если получен сегмент с некоторым кодом поля *номер октета, который должен прийти следующим*, это означает, что все октеты с номерами меньше указанного в данном поле, доставлены благополучно. Если благополучно доставлены байты с номерами 0-N, а затем получен сегмент с номерами байтов (N+k) - (N+k+m), такой сегмент будет буферизован, но подтверждения его получения не последует. Вместо этого посылается отклик, с кодом *номер октета, который должен прийти следующим* =(N+1). В случае получения сегмента с неверной контрольной суммой будет послан отклик, идентичный предыдущему. Дублированные отклики позволяют детектировать потерю пакета.

Поле ***HLEN*** - определяет длину заголовка сегмента, которая измеряется в 32-разрядных словах. Это поле нужно, так как в заголовке могут содержаться поля опций пееременной длины. Далее следует поле ***резерв***, предназначенное для будущего использования, в настоящее время должно обнуляться. Поле ***размер окна*** сообщает, сколько октетов готов принять получатель (флаг ACK=1) вслед за байтом, указанным в поле *номер октета, который должен прийти следующим*. Окно имеет принципиальное значение, оно определяет число сегментов, которые могут быть посланы без получения подтверждения. Значение ширины окна может варьироваться во время сессии (смотри описание процедуры "медленного старта"). Значение этого поля равное нулю также допустимо и указывает, что байты вплоть до указанного в поле *номер октета, который должен прийти следующим*, получены, но адресат временно не может принимать данные. Разрешение на посылку новой информации может быть дано с помощью посылки сегмента с тем же значением поля *номер октета, который должен прийти следующим*, но ненулевым значением поля ширины окна. Поле ***контрольная сумма*** предназначено для обеспечения целостности сообщения. Контрольное суммирование производится по модулю 1. Перед контрольным суммированием к TCP-сегменту добавляется псевдозаголовок, как и в случае протокола [udp](http://book.itep.ru/4/44/udp_442.htm), который включает в себя адреса отправителя и получателя, код протокола и длину сегмента, исключая псевдозаголовок. Поле ***указатель важной информации*** представляет собой указатель последнего байта, содержащий информацию, которая требует немедленного реагирования. Поле имеет смысл лишь при флаге URG=1, отмечающем сегмент с первым байтом "важной информации". Значение разрядов в 6-битовом коде ***флаги*** описано в таблице 4.4.3.1. Если флаг ACK=0, значение поля *номер октета, который должен прийти следующим,* игнорируется. Флаг URG=1 устанавливается в случае нажатия пользователем клавиш Del или Ctrl-С.

Таблица 4.4.3.1 Значения бит поля *флаги*

|  |  |
| --- | --- |
| Обозначение битов (слева на право) поля ***флаги*** | Значение бита, если он равен 1 |
| **URG** | Флаг важной информации, поле *Указатель важной информации* имеет смысл, если urg=1. |
| **ACK** | Номер октета, который должен прийти следующим, правилен. |
| **PSH** | Этот сегмент требует выполнения операции push. Получатель должен передать эти данные прикладной программе как можно быстрее. |
| **RST** | Прерывание связи. |
| **SYN** | Флаг для синхронизации номеров сегментов, используется при установлении связи. |
| **FIN** | Отправитель закончил посылку байтов. |

**Формат TCP-сегмента**

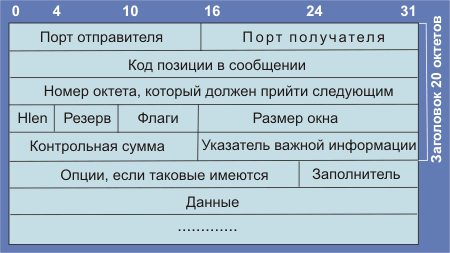


Рис. 4.4.3.1 Формат TCP-сегмента

Поле ***опции*** зарезервировано на будущее и в заголовке может отсутствовать, его размер переменен и дополняется до кратного 32-бит с помощью поля *заполнитель*. Формат поля ***опции*** представлен на рис. 4.4.3.2. В настоящее время определены опции:

0   Конец списка опций.  
1   Никаких операций. Используется для заполнения поля опции до числа октетов, кратного 4.  
2   Максимальный размер сегмента (MSS).

В поле ***вид*** записывается код опции, поле ***LEN*** содержит число октетов в описании опции, включая поля *вид* и *LEN*. Определены также опции со значением вид=4,5,6,7. В предложении T/TCP (RFC-1644) описаны опции 11, 12 и 13. Поле ***данные*** может иметь переменную длину, верхняя его граница задается значением **MSS** (Maximum Segment Size). Значение MSS может быть задано при установлении соединения каждой из сторон независимо. Для Ethernet MSS=1452 байта.

**Формат опций для TCP-сегментов**

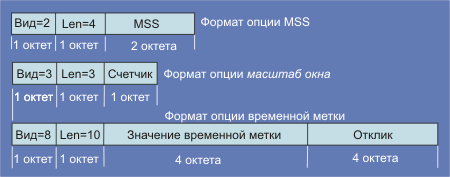


Рис. 4.4.3.2. Формат опций для TCP-сегментов

Поле *данные* в TCP-сегменте может и отсутствовать, характер и формат передаваемой информации задается исключительно прикладной программой, теоретически максимальный размер этого поля составляет в отсутствии опций 65495 байт (на практике, помимо MSS, нужно помнить, например, о значении MTU для Ethernet, которое немногим больше 1500 байт). TCP является протоколом, который ориентируется на согласованную работу ЭВМ и программного обеспечения партнеров, участвующих в обмене информацией. Установление связи клиент-сервер осуществляется в три этапа:

1. Клиент посылает SYN-сегмент с указанием номера порта сервера, который предлагается использовать для организации канала связи (active open).
2. Сервер откликается, посылая свой SYN-сегмент, содержащий идентификатор (**ISN** - Initial Sequence Number). Начальное значение ISN не равно нулю. Процедура называется *passive open*.
3. Клиент отправляет подтверждение получения SYN-сегмента от сервера с идентификатором равным ISN (сервера)+1.

Стандартная процедура установления связи представлена на рисунке 4.4.3.3 (под словом “стандартная” подразумевается отсутствие каких-либо отклонений от штатного режима, например, одновременного открывание соединения со стороны сервера и клиента). Если же соединение одновременно инициируется клиентом и сервером, в конечном итоге будет создан только один виртуальный канал.

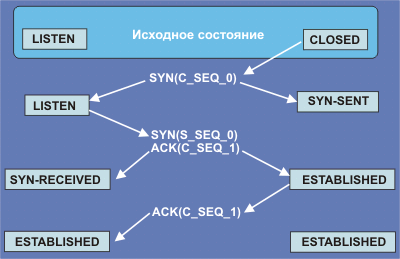


Рис. 4.4.3.3. Алгоритм установления связи. В рамках представлены состояния клиента и сервера; пунктиром отмечены изменения cостояния после посылки сообщения (см. также рис. 4.4.3.4)

Префикс **S** на рисунке указывает на сервер, а **С** - на клиента. Параметры в скобках обозначают относительные значения ISN. После установления соединения ISN(S) = s\_seq\_1, а ISN(C) = c\_seq\_1.

Каждое соединение должно иметь свой неповторимый код ISN. Для реализации режима соединения прикладная программа на одном конце канала устанавливается в режим пассивного доступа ("passive open"), а операционная система на другом конце ставится в режим активного доступа ("active open"). Протокол TCP предполагает реализацию 11 состояний (established, closed, listen, syn\_sent, syn\_received и т.д.; см. также RFC-793), переход между которыми строго регламентирован. Машина состояний для протокола TCP может быть описана диаграммой, представленной на рис. 4.4.3.4. Здесь состояние closed является начальной и конечной точкой последовательности переходов. Каждое соединение стартует из состояния closed. Из диаграммы машины состояний видно, что ни одному из состояний не поставлен в соответствие какой-либо таймер. Это означает, что машина состояний TCP может оставаться в любом из состояний сколь угодно долго. Исключение составляет keep-alive таймер, но его работа является опционной, а время по умолчанию составляет 2 часа. Это означает, что машина состояния может оставаться 2 часа без движения. В случае, когда две ЭВМ (C и S) попытаются установить связь друг с другом одновременно, реализуется режим simultaneous connection (RFC-793). Обе ЭВМ посылают друг другу сигналы SYN. При поучении этого сигнала партнеры посылают отклики SYN+ACK. Обе ЭВМ должны обнаружить, что SYN и SYN+ACK относятся к одному и тому же соединению. Когда C и S обнаружат, что SYN+ACK соответствует посланному ранее SYN, они выключат таймер установления соединения и перейдут непосредственно в состояние syn\_recvd (смотри рис. 4.4.3.4).

В состоянии established пакет будет принят сервером, если его ISN лежит в пределах s\_ack, s\_ack+s\_wind (s\_wind - ширина окна для сервера; см. рис. 4.4.3.5). Аналогичный диапазон ISN для клиента выглядит как: c\_ack, c\_ack+c\_wind (c\_wind - ширина окна для клиента). c\_wind и s\_wind могут быть не равны. *Пакеты, для которых эти условия не выполняются, будут отброшены*.

Рассмотрим пример установления соединения для случая FTP-запроса (См. также [http://www.cis.ohio-state.edu/~dolske/gradwork/cis694q](http://www.cis.ohio-state.edu/%7Edolske/gradwork/cis694q/)). Пусть клиент **С** запускает процесс установления FTP-соединения с сервером **s**. Обычный порядок установления соединения показан ниже (см. рис. 4.4.3.3):

c -> s:syn(ISNc)  
s -> c:syn(ISNs), ack(ISNc)  
c -> s: ack(ISNs) (Связь установлена)  
c -> s: данные  
и/или  
s -> c: данные

ISN - идентификатор пакета, посылаемого клиентом (С) или сервером (S). Клиент, послав SYN серверу S, переходит в состояние *syn\_sent*. При этом запускается таймер установления соединения.

|  |
| --- |
| Как при установлении соединения, так и при его разрыве приходится сталкиваться с проблемой двух армий. Представим себе, что имеется две армии А и Б, причем Б больше по численности чем А. Армия Б разделена на две части, размещенные по разные стороны от армии А. Если две части армии Б одновременно нападут на армию А, победа гарантирована. В то же время нападение на А одной из частей армии Б обрекает ее на поражение. Но как обеспечить одновременность? Здесь предполагается, что радио еще не изобретено и передача сообщений осуществляется вестовыми, которые в нашем случае могут быть перехвачены врагом. Как убедиться, что вестовой дошел? Первое, что приходит в голову, это послать другого вестового с подтверждением. Но он также с некоторой вероятностью может быть перехвачен. А отправитель не будет знать, дошел ли он. Ведь если сообщение перехвачено, отправитель первичного запроса не выдаст команды на начало, так как не уверен, дошло ли его первое сообщение. Возникает вопрос, существует ли алгоритм, который бы гарантировал надежность синхронизации решений путем обмена сообщениями при ненадежной доставке? Повысит ли достоверность увеличение числа обменов между партнерами? Ответом на этот вопрос будет - нет, не существует. В этом читатель, порассуждав логически, может убедиться самостоятельно. Не трудно видеть, что схожие проблемы возникают в любом протоколе, работающем через установление соединения. Чаще всего эта проблема решается путем таймаутов и повторных попыток (это, слава богу, не война и все обходится без людских жертв). |

Сервер, получив **SYN**, откликается посылкой другого SYN. Когда **С** получает SYN от **S** (но не получает ACK, например, из-за его потери или злого умысла), он предполагает, что имеет место случай одновременного открытия соединения. В результате он посылает **syn\_ack**, отключает таймер установления соединения и переходит в состояние syn\_received. Сервер получает syn\_ack от C, но не посылает отклика. Тогда С ожидает получения syn\_ack в состоянии syn\_received. Так как время пребывания в этом состоянии не контролируется таймером, **С** может остаться в состоянии syn\_received вечно. Из-за того, что переходы из состояния в состояние не всегда четко определены, протокол TCP допускает и другие виды атак (некоторые из них описаны в разделе [“Сетевая безопасность”](http://book.itep.ru/6/secur_6.htm)), там же рассмотрены алгоритмы задания и изменения ISN.

Хотя TCP-соединение является полнодуплексным, при рассмотрении процесса разрыва связи проще его рассматривать как два полудуплексных канала, каждый из которых каналов ликвидируется независимо. Сначала инициатор разрыва посылает сегмент с флагом **FIN**, сообщая этим партнеру, что не намерен более что-либо передавать (FIN посылается, как правило в результате вызова приложением функции ***close***). Когда получение этого сегмента будет подтверждено (ACK), данное направление передачи считается ликвидированным (реализуется полузакрытие соединения). При этом передача информации в противоположном направлении может беспрепятственно продолжаться. Когда партнер закончит посылку данных, он также пошлет сегмент с флагом FIN. По получении отклика ACK виртуальный канал считается окончательно ликвидированным.

|  |
| --- |
| Таким образом, для установление связи требуется обмен тремя сегментами, а для разрыва - четырьмя. Но протокол допускает совмещение первого ACK и второго FIN в одном сегменте, сокращая полное число закрывающих сегментов с четырех до трех. |

Партнер, пославший флаг FIN первым, производит *активное закрытие соедиения*, а противоположный партнер (получивший FIN) отвечает на него своим FIN, осуществляя *пассивное закрытие соединения*. Инициатором посылки первого FIN может любая из сторон, но чаще это делается клиентом (например, путем ввода команды quit). Полузакрытие используется, например при реализации команды **rsh** (запуск операций в удаленном узле).

|  |
| --- |
| Машина состояний для протокола TCP не предусматривает изменения состояний при посылке или получении обычных пакетов, содержащих данные. |

Всего в машине конечных состояний протокола TCP имеется 11 состояний (CLOSED, LISTEN, SYN\_RCVD, SYN\_SENT и т.д., введены в RFC-793). Состояние CLOSED является начальной и конечной точкой диаграммы. ESTABLISHED указывает на то, что система находится в состоянии с установленным соединением. Четыре состояния в левом углу помещены в границы зеленой зоны и соответствуют активному закрытию. Состояния CLOSE\_WAIT и LAST\_ACK относятся к пассивному закрытию. Переход из состояния SYN\_RCVD в LISTEN возможно, если переход в SYN\_RCVD осуществлен из состояния LISTEN, а не из состояния SYN\_SENT (одновременное открытие двух соединений, получение RST вместо финального ACK).

**Машина состояний для протокола TCP**

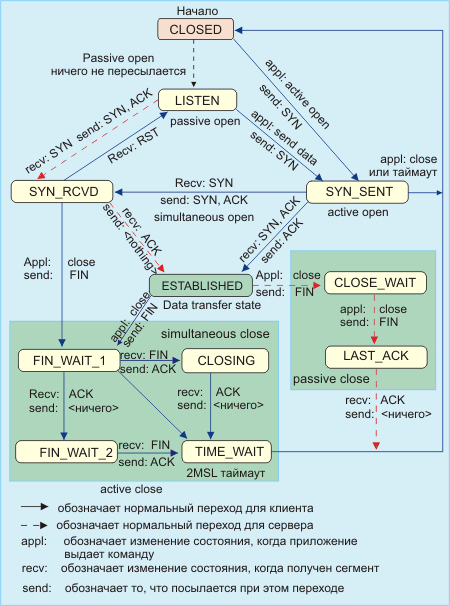


Рис. 4.4.3.4. Машина состояний для протокола TCP (W.R. Stivens, TCP/IP Illustrated. V1. Addison-Wesley publishing company. 1993. Имеется обновленная версия книги, переведенная на русский язык: У.Ричард Стивенс, "Протоколы TCP/IP. Практическое руководство", BHV, Санкт-Петербург, 2003)

Состояние TIME\_WAIT часто называется ожиданием длительностью 2MSL (Maximum Segment Lifetime). Значение MSL задается конкретной реализацией и определяет предельную величину пребывания сегмента в сети. В RFC-793 рекомендуется задавать MSL равным 2 мин. Но нужно помнить, что ТСР-сегмент транспортируется в IP-дейтаграмме, содержащем поле TTL. Когда модуль выполнил активное закрытие и в ответ на FIN послал ACK, соединение должно оставаться в состоянии TIME\_WAIT в течение времени, в два раза превышающем MSL. Сокет, используемый данным соединение не может быть задействован другим соединением в продолжении указанного времени. Все сегменты данного соединения, задержавшиеся в пути, во время TIMR\_WAITотбрасываются. Это гарантирует то, что сегменты старого соединения не будут восприняты новым соедиением. Такая процедура препятствует перезапуску серверов в течение 1-4 минут, так как в течение данного времени не могут использоваться стандартные значения номеров портов.

Состояние FIN\_WAIT\_2 сопряжено со случаем, когда одна сторона послала сегмент FIN, а другая сторона подтвердила его получение. Если данное соединение не нужно, можно ждать, когда приложение другой стороны получит код конца файла и пришлет свой флаг FIN. Только после этого система перейдет из состояний FIN\_WAIT\_2 в состояние TIME\_WAIT. Теоретически такое ожидание может быть бесконечным. Другая сторона при этом остается в состоянии CLOSE\_WAIT, пока приложение не вызовет функцию **close**. Для решения проблемы часто вводят дополнительный таймер.

В ТСР возможна ситуация, когда обе стороны запускают процедуру закрытия одновременно (посылают FIN), что в протоколе ТСР вполне допустимо. Каждая из сторон при этом переходит из состояния ESTABLISHED в состояние FIN\_WAIT\_1 (после вызова операции closed). По получении FIN стороны переходят из состояния FIN\_WAIT\_1 в состояние CLOSING и посылают ACK. После получения ACK происходит переход в состояние TIME\_WAIT.

Когда оператор, работая в диалоговом режиме, нажимает командную клавишу, сегмент, в который помещается эта управляющая последовательность, помечается флагом PSH (push). Это говорит приемнику, что информация из этого сегмента должна быть передана прикладному процессу как можно скорее, не дожидаясь прихода еще какой-либо информации. Сходную функцию выполняет флаг URG. URG позволяет выделить целый массив данных, так как активизирует указатель последнего байта важной информации. Будет ли какая-либо реакция на эту "важную" информацию определяет прикладная программа получателя. URG-режим используется для прерываний при работе с FTP, telnet или rlogin. Если до завершения обработки "важной" информации придет еще один сегмент с флагом URG, значение старого указателя конца "важного" сообщения будет утеряно. Это обстоятельство должно учитываться прикладными процессами. Так telnet в командных последовательностях всегда помещает префиксный байт с кодом 255.

**Алгоритм Нагля**

В режиме удаленного терминала (telnet/ssh) при нажатии любой клавиши формируется и поcылается 41-октетный сегмент (здесь не учитываются издержки Ethernet), который содержит всего один байт полезной информации. В локальной сети здесь проблем не возникает, но в буферах маршрутизаторов в среде Интернет могут возникнуть заторы. Эффективность работы может быть улучшена с помощью алгоритма ***Нагля*** (Nagle, 1984; RFC-896). Нагль предложил при однобайтовом обмене посылать первый байт, а последующие буферизовать до прихода подтверждения получения посланного. После этого посылаются все буферизованные октеты, а запись в буфер вводимых кодов возобновляется. Если оператор вводит символы быстро, а сеть работает медленно, этот алгоритм позволяет заметно понизить загрузку канала. Встречаются, впрочем, случаи, когда алгоритм Нагля желательно отключить, например, при работе в Интернет в режиме Х-терминала, где сигналы перемещения мышки должны пересылаться немедленно, чтобы не ввести в заблуждение пользователя относительно истинного положения маркера.

**Синдром узкого окна**

Существует еще одна проблема при пересылке данных по каналам TCP, которая называется *синдром узкого окна* (silly window syndrome; Clark, 1982). Такого рода проблема возникает в том случае, когда данные поступают отправителю крупными блоками, а интерактивное приложение адресата считывает информацию побайтно. Предположим, что в исходный момент времени буфер адресата полон и передающая сторона знает об этом (window=0). Интерактивное приложение считывает очередной октет из TCP-потока, при этом TCP-агент адресата поcылает уведомление отправителю, разрешающее ему послать один байт. Этот байт будет послан и снова заполнит до краев буфер получателя, что вызовет отправку ACK со значением window=0. Этот процесс может продолжаться сколь угодно долго, понижая коэффициент использования канала ниже паровозного уровня.

Кларк предложил не посылать уведомление о ненулевом значении ширины окна при считывании одного байта, а лишь после освобождения достаточно большого пространства в буфере. Например, когда адресат готов принять MSS байтов или когда буфер наполовину пуст.

Предполагается, что получатель пакета практически всегда посылает отправителю пакет-отклик. Отправитель может послать очередной пакет, не дожидаясь получения подтверждения для предшествующего. Таким образом, может быть послано k пакетов, прежде чем будет получен отклик на первый пакет (протокол "скользящего окна").

|  |
| --- |
| В протоколе TCP "скользящее окно" используется для регулировки трафика и препятствия переполнения буферов. |

Идея скользящего окна отображена на рис. 4.4.3.5. Здесь предполагается, что ширина окна равна 7 (k=7; это число может меняться в очень широких пределах).



Рис. 4.4.3.5. Схема использования скользящего окна

После прихода отклика на пакет <1> окно смещается вправо на одну позицию. Теперь отправитель может послать и пакет <8>. Если порядок прихода откликов нарушается, сдвиг окна может задержаться. Размер окна в сегментах определяется соотношением:

**window > RTT×B/MSS**,

где **B** - полоса пропускания канала в бит/с, а **MSS** - максимальный размер сегмента в битах, а **window** - в сегментах.

Для протокола TCP механизм скользящего окна может работать на уровне октетов или сегментов. В первом случае нужно учитывать каждый раз размер поля данных переданного и подтвержденного сегмента. В TCP-протоколе используется три указателя (стрелки на рис. 4.4.3.3б):

Первый указатель определяет положение левого края окна, отделяя посланный сегмент, получивший подтверждение, от посланного сегмента, получение которого не подтверждено. Второй указатель отмечает правый край окна и указывает на сегмент, который может быть послан до получения очередного подтверждения. Третий указатель помечает границу внутри скользящего окна между уже посланными сегментами и теми, которые еще предстоит послать. Получатель организует аналогичные окна для обеспечения контроля потока данных. Если указатель 3 совпадет с указателем 2, отправитель должен прервать дальнейшее отправление пакетов до получения хотя бы одного подтверждения. Обычно получатель посылает одно подтверждение (ACK) на два полученных сегмента.

Регулирование трафика в TCP подразумевает существование двух независимых процессов: *контроль доставки*, управляемый получателем с помощью параметра **window**, и *контроль перегрузки*, управляемый отправителем с помощью окна перегрузки **cwnd** (congestion window) и **ssthreth** (slow start threshold). Первый процесс отслеживает заполнение входного буфера получателя, второй - регистрирует перегрузку канала, а также связанные с этим потери и понижает уровень трафика. В исходный момент времени при установлении соединения cwnd делается равным одному MSS, а ssthreth=65535 байтам. Программа, управляющая пересылкой, никогда не пошлет больше байт, чем это задано cwnd и объявленным получателем значением window. Когда получение очередного блока данных подтверждено, значение cwnd увеличивается. Характер этого увеличения зависит от того, осуществляется медленный старт или реализуется процедура подавления перегрузки. Если cwnd меньше или равно ssthreth, выполняется медленный старт, в противном случае осуществляется подавление перегрузки. В последнем случае cwndi+1 = cwndi + MSS/8 +(MSS\*MSS)/cwnd. Если возникает состояние перегрузки канала значение cwnd снова делается равным одному MSS.

|  |
| --- |
| Окно перегрузки (CWND) позволяет согласовать полную загрузку виртуального соединения и текущие возможности канала, минимизируя потери пакетов при перегрузке. |

В качестве модуля приращения cwnd используется MSS. При получении подтверждения (ACK) окно перегрузки увеличивается на один сегмент (*"медленный старт"*, CWNDi+1 = CWNDi + размер\_сегмента, последнее слагаемое нужно, если размер окна задан в октетах, в противном случае вместо него следует использовать 1) и теперь отправитель может послать, не дожидаясь ACK, уже два сегмента и т.д.. Ширина окна, в конце концов, может стать настолько большой, что ошибка доставки в пределах окна станет заметной. Тогда будет запущена процедура “медленного старта” или другой алгоритм, который определит новое, уменьшенное значение окна. Окно перегрузки позволяет управлять информационным потоком со стороны отправителя, блокируя возможные перегрузки и потери данных в промежуточных узлах сети (о других методах подавления перегрузки канала смотри [раздел "Сети передачи данных"](http://book.itep.ru/4/net_4.htm#point77)). Если переполнения не происходит, CWND становится больше окна, объявленного получателем, и именно последнее будет ограничивать поток данных в канале. Размер окна, объявленный получателем, ограничивается произведением полосы пропускания канала (бит/с) на RTT (время распространения пакета туда и обратно). Максимально допустимый размер окна в TCP равен 65535 байт (задается размером поля). Конечной целью регулирования трафика является установление соответствия между темпом передачи и возможностями приема. Причиной перегрузки может быть не только ограниченность размера буфера, но и недостаточная пропускная способность какого-то участка канала. С учетом этого обстоятельства каждый отправитель формирует два окна: *окно получателя* и *окно перегрузки* (ширина этого окна равна cwnd). Каждое из этих окон задает число байтов, которое может послать отправитель. Реальное число байтов, которое разрешено послать, равно минимальному из этих окон. При инициализации соединения окно перегрузки имеет ширину равную максимальному сегменту, который может быть использован в данном канале. Отправитель посылает такой сегмент. Если будет прислано подтверждение до истечения времени таймаута, размер окна перегрузки удваивается и посылается два сегмента максимальной длины. При получении подтверждения доставки каждого из сегментов окно перегрузки увеличивается на один сегмент максимальной длины. Когда ширина окна перегрузки становится равной B сегментов и все B посланных сегментов получают подтверждение, окно перегрузки возрастает на число байт, содержащихся в этих сегментах. Таким образом, ширина окна перегрузки последовательно удваивается, пока доставка всех сегментов подтверждается. Рост ширины окна перегрузки при этом имеет экспоненциальный характер. Это продолжается до тех пор, пока не наступит таймаут или окно перегрузки не сравняется с окном получателя. Именно эта процедура и называется ***медленным стартом*** (Джекобсон, 1988).

Как было сказано выше, помимо окон перегрузки и получателя в TCP используется третий параметр - *порог* (иногда он называется порогом медленного старта ssthresh). При установлении соединения ssthresh=64 Kбайт. В случае возникновения таймаута значение **ssthresh** делается равным **CWND/2**, а само значение CWND приравнивается MSS (см. рис. 4.4.3.6). Далее запускается процедура медленного старта, чтобы выяснить возможности канала. При этом экспоненциальный рост **cwnd** осуществляется вплоть до значения **ssthresh**. Когда этот уровень **cwnd** достигнут, дальнейший рост происходит линейно с приращением на каждом шагу равным **MSS** (рис. 4.4.3.6).

**Эволюция ширины окна при медленном старте**

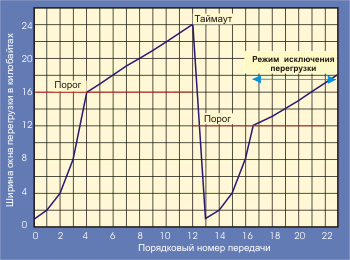


Рис. 4.4.3.6. Эволюция ширины окна при *медленном старте*

Здесь предполагается, что MSS=1 Кбайт. Началу диаграммы соответствует установка значения ssthresh=16 Kбайт. Данная схема позволяет более точно выбрать значение **cwnd**. После таймаута, который на рисунке произошел при передаче c номером 12, значение порога понижается до 12 Кбайт (=cwnd/2). Ширина окна cwnd снова начинает расти от передачи к передаче, начиная с одного сегмента, вплоть до нового значения порога ssthresh=12 Кбайт. Стратегия с экспоненциальным и линейным участками изменения ширины окна переполнения позволяет несколько приблизить среднее его значение к оптимальному. Для локальных сетей, где значение RTT невелико, а вероятность потери пакета мала, оптимизация задания **cwnd** не так существенна, как в случае протяженных внешних (например, спутниковых) каналов. Ситуация может поменяться, если в локальной сети имеется фрагмент, где вероятность потерь пакетов велика. Таким фрагментом может быть МАС-бридж (или переключатель), один из каналов которого подключен к сегменту Fast Ethernet, а другой к обычному Ethernet на 10 Мбит/c. Если такой мост не снабжен системой подавления перегрузки (до сих пор такие приборы не имели подобных систем), то каждый из пакетов будет потерян в среднем 9 раз, прежде чем будет передан (здесь предполагается, что передача идет из сегмента FE). При этом cwnd будет практически все время равно MSS, что крайне неэффективно при передаче по каналам Интернет. Такие потери вызовут определенные ошибки при вычислении среднего значения и дисперсии RTT, а как следствие и величин таймаутов. Применение в таких местах маршрутизаторов или других приборов, способных реагировать на перегрузку посредством **ICMP(4)**, решает эту проблему.

**Таймеры ТСР**

Для взаимного согласования операций в рамках TCP-протокола используется четыре таймера:

1. Таймер повторных передач (retransmission; RTO) контролирует время прихода подтверждений (**ACK**). Таймер запускается в момент посылки сегмента. При получении отклика ACK до истечения времени таймера, он сбраcывается. Если же время таймера истекает до прихода ACK, сегмент посылается адресату повторно, а таймер перезапускается.
2. Таймер запросов (**persist timer**), контролирующий размер окна даже в случае, когда приемное окно закрыто. При window=0 получатель при изменении ситуации посылает сегмент с ненулевым значением ширины окна, что позволит отправителю возобновить свою работу. Но если этот пакет будет потерян, возникнет тупик, тогда каждая из сторон ждет сигнала от партнера. Именно в этой ситуации и используется таймер запросов. По истечении времени этого таймера отправитель пошлет сегмент адресату. Отклик на этот сегмент будет содержать новое значение ширины окна. Таймер запускается каждый раз, когда получен сегмент с window=0.
3. Таймер контроля работоспособности (**keepalive**), который регистрирует факты выхода из строя или перезагрузки ЭВМ-партнеров. Время по умолчанию равно 2 часам. Keepalive-таймер не является частью TCP-спецификации. Таймер полезен для выявления состояний сервера half-open при условии, что клиент отключился (например, пользователь выключил свою персональную ЭВМ, не выполнив LOGOUT). По истечении времени таймера клиенту посылается сегмент проверки состояния. Если в течение 75 секунд будет получен отклик, сервер повторяет запрос 10 раз с периодом 75 сек, после чего соединение разрывается. При получении любого сегмента от клиента таймер сбрасывается и запускается вновь.
4. **2MSL**-таймер (Maximum Segment Lifetime) контролирует время пребывания канала в состоянии TIME\_WAIT. Выдержка таймера по умолчанию равно 2 мин (**FIN\_WAIT**-таймер). См. рис. 4.4.3.4. и RFC-793. Таймер запускается при выполнении процедуры active close в момент посылки последнего ACK.

Важным параметром, определяющим рабочие параметры таймеров, является RTT (время путешествия пакета до адресата и обратно). TCP-агент самостоятельно измеряет RTT. Такие измерения производятся периодически и по их результатам корректируется среднее значение RTT:

**RTTm = a×RTTm + (1-a)×RTTi**,

где **RTTi** - результат очередного измерения, **RTTm** - величина, полученная в результате усреднения предыдущих измерений, **а** - коэффициент сглаживания, обычно равный 0.9. RFC-793 рекомендует устанавливать время таймаута для ретрансмиссии (повторной передачи), значение **RTO** - Retransmission TimeOut равно RTO=RTTm\*b, где **b** равно 2. От корректного выбора этих параметров зависит эффективная работа каналов. Так занижение времени ретрансмиссии приводит к неоправданным повторным посылкам сегментов, перегружая каналы связи. Для более точного выбора RTO необходимо знать дисперсию RTT. Несколько более корректную оценку RTO можно получить из следующих соотношений (предложено Джекобсоном в 1988 году, он же позднее предложил целочисленный алгоритм реализации этих вычислений):

**RTTm = RTTm + g(RTTi-RTTm)  
D = D + d(|RTTi - RTTm| - D)  
RTO = RTTm + 4D,**

где **D** - среднее отклонение RTT от равновесного значения, а коэффициенты **g** = 0,125, **D** = 0,25. Чем больше **g**, тем быстрее растет **RTO** по отношению к RTT. Это хорошо работает до тех пор, пока не произойдет таймаут и ретрансмиссия. В этом случае, получив ACK, трудно решить, какому сегменту соответствует это подтверждение, первому или второму. На эту проблему впервые обратил внимание Фил Карн. Решением проблемы является приостановка коррекции **RTTm** при таймауте и ретрансмиссиях. Значение RTO зависит от пропускной способности канала и от специфических задержек, например в случае спутниковых каналов. В основном RTO лежит в секундном диапазоне (5-15 сек). Наиболее вероятная причина потери пакетов - это перегрузка канала на участках между отправителем и приемником. Указанием на то, что пакет потерян, может служить таймаут или получение дубликата сегмента ACK. Если произошел таймаут, система переходит в режим "медленного старта" (ширина окна перегрузки делается равной 1 сегменту, а значение порога медленного старта - **ssthresh** делается равным двум сегментам). При инициализации канала переменная **ssthresh** обычно равна 65535. Дублирование ACK индицирует потерю пакета до наступления таймаута. В этом случае сначала меняется алгоритм приращения величины окна перегрузки **cwnd** (замедляется темп его роста). После прихода очередного ACK новое значение cwnd вычисляется по формуле:

**cwndi+1 = cwndi + (размер\_сегмента×размер\_сегмента)/cwndi + размер\_сегмента/8**

Если же в этот момент величина окна перегрузки меньше или равна некоторому порогу (**ssthresh** - slow start threshold, обычно измеряется в байтах), осуществляется "медленный старт". Следует помнить, что TCP требует посылки немедленного подтверждения (дублированного ACK) при обнаружении прихода сегментов с нарушением порядка следования. Причиной нарушения порядка следования может быть флуктуация задержки в сети или потеря пакета. Если получено три или более задублированных ACK, это является убедительным указанием на потерю пакета и, не дожидаясь таймаута, осуществляется его повторная передача. Перехода в режим медленного старта в этом случае не производится, но понижаются значения **cwnd** и **ssthresh** (почти вдвое).

Когда TCP-канал закрывается и за время сессии переслано более 16 полых окон, а адресат достижим не через маршрут по умолчанию, то в таблицу маршрутизации заносится следующая информация: усредненное значение RTT, значение дисперсии **RTT** и **ssthresh**.

Если в ходе TCP-сессии получено сообщение **ICMP(4)** (переполнение канала - quench), требующее снижения потока данных, то cwdn делается равным одному сегменту, а величина порога медленного старта ssthresh не изменяется. На ICMP-сообщения о недостижимости сети или ЭВМ программы TCP-уровня не реагируют вообще.

Нулевой размер окна блокирует посылку информации и этим система время от времени пользуется. Что произойдет, если получатель послал сегмент, объявляющий окно ненулевым, а подтверждение получения этого сегмента не прошло? TCP-протокол не предусматривает посылки ACK на само подтверждение. Адресат ждет в этом случае данных, так как он уже объявил о существовании ненулевого окна с помощью соответствующего ACK, а отправитель ждет этого недошедшего ACK, чтобы начать передачу данных. Для разрешения этой тупиковой ситуации используется таймер запросов, который периодически посылает зондирующие сегменты получателю. Цель этого зондирования - выяснение существования окна ненулевой ширины. Таймер запросов запускается при получении информации об обнулении ширины окна приемником. Если за определенное время не поступает сегмента, сообщающего об изменении размера окна, таймер начинает посылать зондирующие сегменты. Таймер запросов использует базовую временную шкалу с периодом в 500 мсек, а период посылки зондирующих сегментов лежит в диапазоне 5-60 сек. Такой сегмент содержит только один байт данных. Таймер запросов не прерывает своей работы до тех пор, пока не будет подтверждено открытие окна или пока прикладная задача не завершит свою работу, выключив канал связи.

Будучи однажды создан, канал TCP может существовать "вечно". Если клиент и сервер пассивны, они не заметят того, например, что какой-то бульдозер оборвал кабель или спутник связи покоится на дне океана. Чтобы это обнаружить, либо клиент либо сервер должны попытаться послать какую-то информацию. Чтобы информировать систему об этих и подобных им жизненных неурядицах, предусмотрен таймер контроля работоспособности (keepalive). Многим читателям, возможно, приходилось легкомысленно выключать питание своего персонального компьютера, не позаботившись о корректном logout из процедуры telnet или FTP. Если бы не существовало этого таймера, включив ЭВМ, вы бы обнаружили, что "находитесь" в заморском депозитарии, где были вчера. Но таймер контроля работоспособности может и прервать сессию, если какой-то промежуточный маршрутизатор произвел перезагрузку или был вынужден поменять маршрут. Принцип работы таймера работоспособности предельно прост. Если канал пассивен, например, 2 часа, сервер посылает клиенту сегмент-зонд. При этом ЭВМ-клиент может быть в одном из четырех состояний.

* Работоспособен и достижим для сервера. Отклик от клиента сбросит таймер работоспособности в ноль (начало отсчета очередных двух часов).
* Вышел из строя, выключен или перезагружается. Сервер посылает 10 запросов с интервалом 75 сек. Если отклика нет, канал закрывается и со стороны сервера.
* Перезагрузился. Сервер получит отклик типа RESET и канал будет закрыт.
* Работоспособен, но не достижим для сервера. Случай тождественен, описанному во втором по порядку пункте.

Временная постоянная таймера keepalive является системной переменной единой для всех пользователей ЭВМ или даже локальной сети.

Расширение пропускной способности и надежности телекоммуникационных каналов делает актуальной совершенствование протоколов. Так как TCP является основным транспортным протоколом, попытки усовершенствовать его предпринимаются, начиная с 1992 года (RFC-1323, Якобсон, Браден и Борман). Целью этих усовершенствований служит повышение эффективности и пропускной способности канала, а также обеспечение безопасности. При этом рассматриваются следующие возможности:

* увеличение MTU (максимальный передаваемый блок данных);
* расширение окна за пределы 65535 байт;
* исключение "трех-сегментного" процесса установления связи и "четырехсегментного" ее прерывания (T/TCP, RFC-1644);
* совершенствование механизма измерения RTT.
* оптимизация отслеживания CWND.

Оптимальный выбор MTU позволяет минимизировать или исключить фрагментацию (и последующую сборку) сегментов. Верхняя граница на MTU налагается значением MSS (максимальный размер сегмента). Разумно находить и запоминать оптимальные значения MTU для каждого конкретного маршрута. Так как в современных системах используются динамические протоколы маршрутизации, поиск оптимального MTU рекомендуется повторять каждые 10 мин (RFC-1191).

Как уже отмечалось, размер TCP-окна определяется произведением полосы канала (в бит/с) на RTT в сек. Для Ethernet c полосой 10 Мбит/с и RTT=3 мсек это произведение равно 3750 байт, а для канала ИТЭФ-ДЕЗИ с пропускной способностью 1,5 Мбит/с и RTT=710 мсек (спутник) - 88750 байт, а это отнюдь не предел современной телекоммуникационной технологии. Но уже эти примеры говорят о том, что максимально возможный размер окна должен быть увеличен в раз 10-100 уже сегодня. Протокол же разрешает 65535 байт. Появление столь мощных каналов порождает и другие проблемы - потеря пакетов в них обходится слишком дорого, так как "медленный старт" и другие связанные с этим издержки сильно снижают пропускную способность. В последнее время алгоритм медленного старта заменяется более эффективными алгоритмами.

Простое увеличение ширины окна до тех пор, пока не произойдет сбой, плохая стратегия при использовании традиционного медленного старта, так как заметную часть времени ширина окна будет неоптимальной - то слишком большой, то слишком малой. Оптимальная стратегия должна включать в себя прогнозирование оптимальной ширины окна. В новых версиях модулей TCP реализуются именно такие алгоритмы. В 1994 году Бракмо предложил вариант стратегии изменения параметров передачи, который на 40-70% повышает пропускную способность TCP-канала.

**Некоторые проблемы ТСР**

Существуют и другие, могущие показаться забавными проблемы. Каждый сегмент в TCP-протоколе снабжается 32-битным идентификатором. Время жизни IP-пакета (TTL) определяется по максимуму 255 шагами или 255 секундами в зависимости оттого, что раньше наступит. Трудно предсказуемая ситуация может произойти, когда канал ликвидирован, затем создан снова (для той же комбинации IP-адресов и портов), а какой-то пакет из предшествующей сессии, погуляв по Интернет, придет уже во время следующей. Есть ли гарантия, что он будет верно идентифицирован? Одной из мер, упомянутых ранее, можно считать использование ограничения по максимальному времени жизни сегмента (**MSL**) или TTL, хотя снижение значения TTL не всегда возможно - ведь IP-пакетами пользуется не только TCP-протокол и нужна очень гибкая система задания его величины. Во многих приложениях MSL=30 сек (рекомендуемое значение 2 мин слишком велико). Технический прогресс ставит и некоторые новые проблемы. Высокопроизводительные каналы (1 Гбит/с) уже сегодня могут исчерпать разнообразие идентификационных кодов пакетов за один сеанс связи. Появление же двух пакетов с равными идентификаторами может породить неразрешимые трудности. Для передачи мегабайтного файла по гигабитному каналу требуется около 40 мсек (при этом предполагается, что задержка в канале составляет 32 мсек (RTT=64 мсек)). Собственно передача этой информации занимает 8 мсек. Из этих цифр видно, что традиционные протоколы, размеры окон и пр. могут свести на нет преимущества скоростного (дорогостоящего) канала. Пропускная способность такого канала определяется уже не его полосой, а задержкой. Понятно также, что необходимо расширить поле размера окна с 16 до 32 бит. Чтобы не изменять формат TCP-сегментов, можно сделать код размера окна в программе 32-разрядным, сохранив соответствующее поле в сегменте неизменным. Размер окна в этом случае задается как бы в формате с плавающей запятой. При установлении канала определяется масштабный коэффициент n (порядок) лежащий в интервале 0-14. Передача этого коэффициента (один байт) осуществляется сегментом SYN в поле опций. В результате размер окна оказывается равным 65535\*2n. Если один из партнеров послал ненулевой масштабный коэффициент, но не получил такого коэффициента от своего партнера по каналу, то **n** считается равным нулю. Эта схема позволит сосуществовать старым и новым системам. Выбор **n** возлагается на TCP-модуль системы.

Для того чтобы точнее отслеживать вариации RTT, предлагается помещать временные метки в каждый посылаемый сегмент. Так как в TCP используется одно подтверждение ACK на несколько сегментов, правильнее будет сказать, что RTT измеряется при посылке каждого ACK. Способность и готовность партнеров работать в таком режиме временных меток определяется на фазе установления канала. Более точное вычисление RTT позволяет не только корректно выбрать временные постоянные для таймеров, правильно вычислить задержку TIME\_WAIT (TIME\_WAIT=8×RTO), но и отфильтровать "старые" сегменты. Идеология временных меток используется и в алгоритме **PAWS** (Protection Against Wrapped Sequence Numbers) для защиты против перепутывания номеров сегментов.

Предлагаемое усовершенствование TCP - T/TCP модифицирует алгоритмы выполнения операций. T/TCP вводит новую 32-битную системную переменную - число соединений (CC). СС позволяет сократить число пересылаемых сегментов при установлении канала, а также отфильтровывать "старые" сегменты, не принадлежащие данной сессии (установленной связи). Время отклика клиента в рамках указанных алгоритмов сокращается до суммы RTT и времени обработки запроса процессором. Данные пришедшие до SYN-сегмента должны буферизоваться для последующей обработки, а не отбрасываться.

Ethernet (10 Мбит/c) в идеальных условиях позволяет осуществить обмен в рамках протокола TCP (например, при FTP-сессии) со скоростью 1,18 Мбайт/с.

Как уже отмечалось, максимальная длина сегмента (MSS - Maximum Segment Size) в TCP-обменах величина переменная. Длина сегмента определяет длину кадра, в который он вложен. Для локальных Ethernet-сетей MSS=1460 октетов. Чем длиннее кадр, тем выше пропускная способность сети (меньше накладные расходы на заголовок кадра). С другой стороны, при передаче дейтограмм по внешним каналам, где размер пакета не столь велик, большое значение MSS приведет к фрагментации пакетов, которая замедлит обмен, поэтому администратор сети должен взвешивать последствия, задавая значения размера сегментов. Если MSS явно не задан, устанавливается значение по умолчанию (536 байт), что соответствует 576-байтной IP-дейтограмме. Для нелокальных адресов - это, как правило, разумный выбор.

Ликвидация связи требует посылки четырех сегментов. TCP-протокол допускает возможность, когда один из концов канала объявляет о прекращении посылки данных (посылает FIN-сегмент), продолжая их получать (режим частичного закрытия - *half-close*). Посылка сегмента FIN означает выполнение операции *active close*. Получатель FIN-сегмента должен послать подтверждение его получения. Когда противоположный конец, получивший FIN, закончит пересылку данных, он пошлет свой FIN-сегмент. Прием подтверждения на получение этого сегмента означает закрытие данного канала связи. Возможно прерывание связи и с помощью посылки RST-сегмента. В этом случае все буферы и очереди очищаются немедленно и часть информации будет потеряна.