Московский государственный технический университет

имени Н.Э.Баумана

Кафедра САПР

Федорук В.Г.

Протоколы сетевого взаимодействия TCP/IP

Москва

Содержание

Литература

1. Протоколы информационно-вычислительных сетей. Под. ред. Мизина И.А. и Кулешова А.П. М.: Радио и связь, 1990, 504 с.
2. Halsall F. Data communications, computer networks and open systems. Addison-Wesley publishing company, 1992, 772 pp.
3. Santifaller M. TCP/IP and ONC/NFS: internetworking in a UNIX environment. Addison-Wesley (Deutschland) GmbH, 1994, 288 pp.

Введение

Протоколы сетевого взаимодействия TCP/IP являются результатом эволюционного развития протоколов глобальной вычислительной сети ARPANET.

Работы по созданию сети ARPANET были начаты рядом университетов США и фирмой BBN в 1968 г. В 1971 г. сеть была введена в регулярную эксплуатацию и обеспечивала для всех своих узлов три основные услуги:

- интерактивный вход пользователя на удаленный узел;
- передача файлов между узлами сети;
- электронная почта.

Все эти средства базировались на транспортных услугах предоставляемых программой управления сети NCP (Network Control Program), реализующей свой внутренний набор протоколов.

Накопленный к 1974 г. опыт эксплуатации сети ARPANET выявил многие недостатки протоколов NCP и позволил определить основные требования к новому набору протоколов, получившему название TCP/IP:

- независимость от среды передачи сообщений;
- возможность подключения к сети ЭВМ любой архитектуры;
- единый способ организации соединения между узлами в сети;
- стандартизация прикладных протоколов.

Широко используемая ныне версия 4 протоколов TCP/IP была стандартизирована в 1981 г. в виде документов, называемых RFC (Request For Comment). Полный переход сети ARPANET на новые протоколы был завершен в 1982 г. Эта сеть сыграла роль "зародыша" всемирной сети Internet, построенной на базе протоколов TCP/IP.

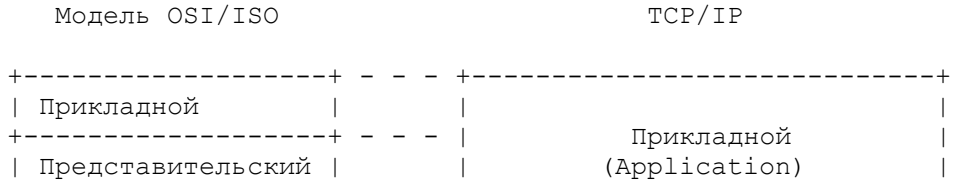
Реализация протоколов TCP/IP оказалась наиболее удачной в версиях BSD4.2 и BSD4.3 операционной системы UNIX. Эта реализация является эталоном (стандартом "de facto") для всех последующих.

Примечание. *Первичным сервером хранения всех RFC является узел* ***nisc.sri.com*** *(доступ через анонимный FTP).*

1. Соотношение между OSI/ISO и TCP/IP

В 1984 г. международная стандартизирующая организация ISO предложила модель взаимодействия открытых систем OSI (Open System Interconnection), являющуюся удобным средством описания стеков протоколов.

На рис. 1.1 представлено соотношение четырехуровневой архитектуры протоколов TCP/IP и семиуровневой архитектуры OSI.



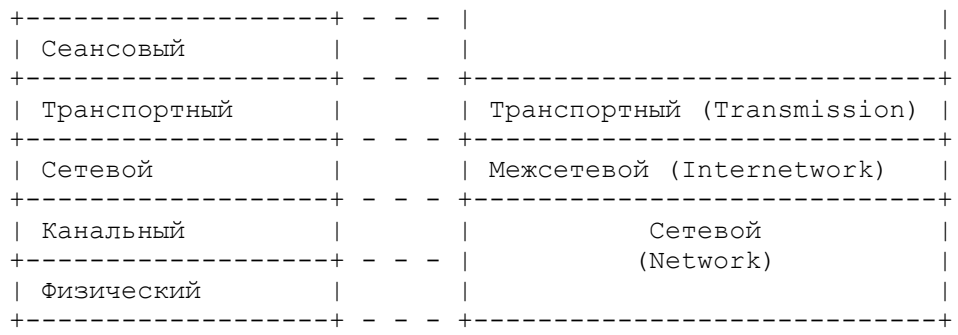


Рис. 1.1

Объединение канального и физического уровней модели OSI в единый сетевой уровень TCP/IP было обусловлено требованием независимости от используемой среды передачи данных. Дело в том, что функции протоколов канального и физического уровней реализуются в настоящее время , как правило, едиными техническими средствами (сетевыми контроллерами).

Согласно терминологии TCP/IP элементы сетевого уровня называются подсетями (subnetworks). Идеология TCP/IP допускает, чтобы в качестве "подсетей" выступали реальные сети с их собственными стеками протоколов, узлами, шлюзами и т.п.

Внимание. *Далее в данном учебном пособии для обозначения уровней стека протоколов используется терминология TCP/IP, а не OSI/ISO (если это не оговорено особо).*

Внимание. *В данном учебном пособии термин "шлюз" используется как обобщающий для понятий "маршрутизатор" (router), "мост" (bridge) и, собственно, "шлюз" (gateway).*

2. Архитектура протоколов TCP/IP

На рис. 2.1 представлена архитектура основных протоколов TCP/IP, используемых на трех нижних уровнях стека.

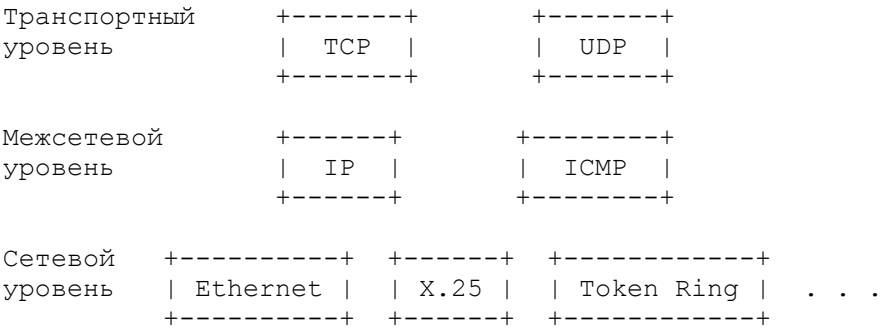


Рис. 2.1

Краеугольным камнем всей архитектуры является межсетевой протокол **IP** (**Internet Protocol**). С его помощью реализуется адресация узлов сети и доставка данных. Межсетевой протокол управляющих сообщений **ICMP** (**Internet Control Message Protocol**) предназначен для передачи диагностической информации и сообщений об ошибках в работе сети.

Примечание. *Протокол ICMP отнесен к межсетевому уровню условно, т.к., с одной стороны, он пользуется возможностями протокола IP для транспортировки*

собственных данных, но, с другой стороны, сам для транспортировки данных пользователя не применяется.

Двумя основными протоколами транспортного уровня являются надежный протокол управления передачей данных **TCP** (**Transmission Control Protocol**) и быстрый протокол дэйтаграмм пользователя **UDP** (**User Datagram Protocol**). TCP реализует сетевое взаимодействие в режиме с установлением логического (виртуального) соединения, а UDP - без оного.

Функции каждого протокола реализуются компонентой программного обеспечения (обычно входящей в состав операционной системы), которую будем называть **модулем**. Взаимодействие модулей соседних уровней осуществляется через стандартизированный интерфейс, имеющий, как правило, процедурный характер.

Внимание. *На каждом уровне стека протоколов TCP/IP обмен данными ведется блоками данных конечной длины. К сожалению, отсутствует устоявшаяся терминология в обозначении этих блоков. В данном учебном пособии названия блоков данных зависят от уровня стека протоколов, как это показано ниже.*

Уровень	Название
Транспортный	Пакет
Межсетевой	Сегмент
Сетевой	Кадр

3. Межсетевой протокол IP

Межсетевой протокол IP специфицирован в RFC 791. Его основные характеристики перечислены ниже:

- реализует обмен информации пакетами, которые будем называть IP-сегментами (максимальный размер IP-сегмента - 65535 байт);
- является протоколом взаимодействия без установления логического соединения;
- для адресации узлов сети используется адрес длиной 4 байта;
- обеспечивает в случае необходимости фрагментацию IP-сегментов;
- IP-сегменты имеют конечное время жизни в сети;
- не гарантирует надежность доставки IP-сегментов адресату;
- не имеет средств управления интенсивностью передачи IP-сегментов посылающей стороной (flow control);
- не гарантирует правильную последовательность IP-сегментов на принимающей стороне.

3.1. Заголовок IP-сегмента

На рис. 3.1 приведен формат заголовка IP-сегмента.

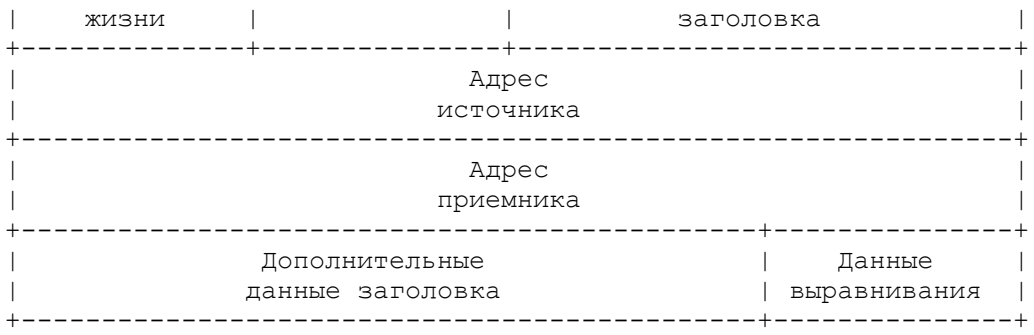


Рис. 3.1

Версия

4-хбитовое поле, содержащее номер версии протокола IP (номер текущей версии равен 4);

Длина заголовка

4-хбитовое поле, содержащее длину заголовка IP-сегмента в 32-битных словах. Минимальная (и типичная) длина заголовка - пять слов.

Тип обслуживания

байт, содержащий набор критериев, определяющих тип обслуживания IP-сегментов. Детальное описание отдельных битов дано ниже:

- биты 0...2 - приоритет (precedence - предпочтение) данного IP-сегмента;
- бит 3 - требование ко времени задержки (delay) передачи IP-сегмента (0 - нормальная, 1 - низкая задержка);
- бит 4 - требование к пропускной способности (throughput) маршрута, по которому должен отправляться IP-сегмент (0 - низкая, 1 - высокая пропускная способность);
- бит 5 - требование к надежности (reliability) передачи IP-сегмента (0 - нормальная, 1 - высокая надежность);
- биты 6...7 - зарезервированы.

На практике в большинстве реализаций протокола IP данное поле почти всегда равно 0, в UNIX-реализациях это поле не используется вовсе.

Длина сегмента

двухбайтовое поле, содержащее длину (в байтах) всего IP-сегмента, включая длину заголовка. Максимальная длина IP-сегмента (включая заголовок) - 65535 байт. Спецификация IP протокола устанавливает, что что любой узел сети должен быть способен обрабатывать IP-сегменты длиной, по крайней мере, не менее 576 байт (что соответствует 512 байтам данных при возможной длине заголовка до 64 байт). На практике же узлы сети могут обрабатывать IP-сегменты много длинее, чем 576 байт (как правило, допустимая длина IP-сегмента связана с максимальной длиной кадра нижележащего сетевого уровня).

Идентификатор

двухбайтовое поле, содержащее уникальный идентификатор IP-сегмента, присваиваемый ему посылающим узлом. Это поле используется для распознавания фрагментов одного IP-сегмента (в ситуациях, когда в ходе перемещения по глобальной сети единый IP-сегмент был разбит на несколько фрагментов по причине его недопустимо большой длины).

DF, MF

биты, используемые при обработке фрагментированных IP-сегментов.

Если бит DF (Don't Fragment) установлен в 1, то это означает, что IP-сегмент не может быть разбит на фрагменты ни при каких условиях (даже, если он не может быть передан без этого далее к адресату и должен быть уничтожен).

Бит MF (More Fragments) указывает, является (MF=0) или нет (MF=1) данный IP-"подсегмент" последним в цепочке IP-"подсегментов", в которую был преобразован (фрагментирован) исходный IP-сегмент.

Алгоритм фрагментации описан ниже в разделе "Фрагментация IP-сегментов".

Смещение фрагмента

13-битное поле, используемое только в IP-сегменте, являющемся фрагментом (IP-фрагментом) другого (исходного) IP-сегмента. Это поле содержит смещение данных, содержащихся в IP-фрагменте, по отношению к началу данных исходного IP-сегмента. Смещение измеряется в восьмибайтных единицах, поэтому 13 битов достаточно для представления смещения в IP-сегменте максимальной возможной длины (8 * 2^13 - 1 = 65535).

Время жизни

однобайтовое поле, заполняемое создающим IP-сегмент узлом сети количеством единиц времени жизни IP-сегмента в сети. RFC 791 специфицирует в качестве этих единиц секунды и требует, чтобы каждый транзитный узел сети, через который проходит IP-сегмент, уменьшал содержимое этого поля по крайней мере на 1 (даже при условии, что обработка сегмента на самом деле заняла меньше одной секунды). Таким образом, на практике, время жизни (**TTL** - Time To Live) - это максимальное количество узлов, которое может пройти до своего уничтожения IP-сегмент.

Каждый IP-модуль на любом узле сети обязан уничтожать IP-сегменты, для которых поле "время жизни" стало равным нулю. Этим предотвращается появление в сети IP-сегментов, "блуждающих" по ней бесконечное время. При этом узлу-источнику уничтоженного IP-сегмента посылается ICMP-сегмент, извещающий об этом событии.

В UNIX-реализациях, как правило, это поле заполняется источником IP-сегмента числом из диапазона 15...30.

Транспорт

поле размером в байт, содержащее идентификатор протокола более высокого (обычно, транспортного) уровня, для которого предназначены данные IP-сегмента. Ниже приведены идентификаторы для ряда протоколов.

Идентификатор	Сокращенное название	Имя протокола
1	ICMP	Межсетевой протокол управляющих сообщений
2	IGMP	Межсетевой протокол группового управления
3	GGP	Протокол "шлюз-шлюз"
6	TCP	Протокол управления передачей
8	EGP	Протокол "внешних" шлюзов
17	UDP	Протокол дейтаграмм пользователя
27	RDP	Протокол надежных данных
28	IRTP	Протокол межсетевой надежной передачи
29	ISO TP4	Транспортный протокол ISO 4 класса
80	ISO IP	Межсетевой протокол ISO
89	OSPF	Протокол "кратчайший путь первым"

Контрольная сумма заголовка

двухбайтовое поле, содержащее контрольную сумму заголовка IP-сегмента (обращаем внимание, что для данных IP-сегмента контрольная сумма не подсчитывается; контролировать данные - задача протоколов транспортного уровня).

Поскольку заголовок IP-сегмента содержит поле "время жизни", изменяющее свое значение в каждом узле, через который следует IP-сегмент, то для вычисления контрольной суммы должен использоваться эффективный (а, следовательно, простой алгоритм). Во всех протоколах, входящих в архитектуру TCP/IP, используется так называемая Internet-контрольная сумма, которая представляет собой дополнение 16-битной суммы всех 16-битных слов контролируемой информации.

Адрес источника и адрес приемника

четырехбайтовые IP-адреса узлов сети. Подробно структура IP-адреса описана ниже в "IP-адрес".

Дополнительные данные заголовка

последовательность полей произвольной длины, описывающих необязательные данные заголовка. Такие данные используются для специальных целей (управление сетью, секретность и т.п.) и кратко описаны ниже в "Дополнительные данные IP-заголовка".

Данные выравнивания

не имеющие смысла данные, включаемые в заголовок только для выравнивания его длины до границы четырехбайтового слова.

3.2. IP-адрес

IP-адрес представляет собой четырехбайтовое число, старшие (крайние левые) биты которого определяют класс IP-адреса. Для классов **A**, **B** и **C** четыре байта адреса делятся между идентификатором (номером) сети и идентификатором (номером) узла в сети как это показано на рис. 3.2.

Рис. 3.2

Сети классов A, B и C абсолютно равноправны и отличаются лишь допустимым количеством узлов в них. Идентификаторы узлов, состоящие из одних нулевых или единичных битов имеют специальный смысл:

- IP-адрес с нулевым идентификатором узла используется для обозначения сети в целом;
- IP-адрес с идентификатор узла в виде единичных битов является **широковещательным (broadcast)** адресом.

IP-адреса принято записывать в так называемой "точечной нотации" - в виде последовательности разделенных точками четырех десятичных (или шестнадцатиричных с префиксом 0x) чисел, представляющих значения отдельных байтов.

Каждый узел в сети имеет, по крайней мере, один уникальный IP-адрес.

Кроме классов A, B и C существуют еще два класса IP-адресов - **D** и **E** (см. рис. 3.3).

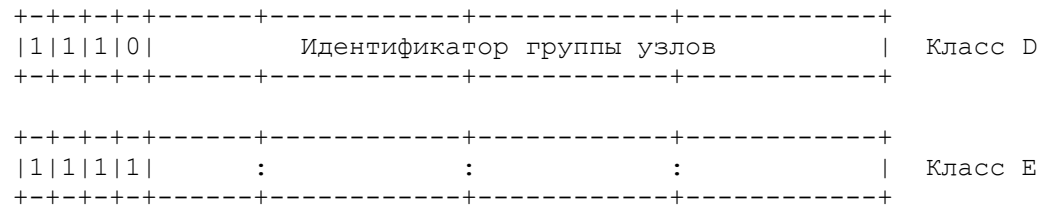


Рис. 3.3

Класс D используется для организации **многопунктового** (**multicast**) режима посылки сообщений: IP-сегмент, посылаемый по по IP-адресу класса D, доставляется всем узлам

сети, имеющим указанный идентификатор группы узлов. Описание данного режима дано в RFC 1112.

Примечание. *Не все современные реализации протоколов TCP/IP поддерживают многопунктовое вещание.*

Для обеспечения гибкости при создании и администрировании сетей различного размера в 1985 г. было введено понятие "подсеть" (RFC 950), позволяющее использовать один и тот же IP-адрес классов A,B или C для разных подсетей.

Такая возможность обеспечивается специальной **битовой маской (netmask)**, ассоциированной с IP-адресом и определяющей распределение битов IP-адреса между идентификатором подсети и идентификатором узла.

Пусть, например, IP-адрес класса C 194.85.36.0 планируется использовать для организации четырех подсетей. Это потребует выделения двух битов из части IP-адреса, относящейся к идентификатору узла. Такое "перепланирование" структуры IP-адреса реализуется сетевой маской 255.255.255.192, где десятичное 192 - это двоичное 11000000.

Эта сетевая маска формирует IP-адрес не из двух, а из трех комронент:

- идентификатор сети (24 бита);
- идентификатор подсети (2 бита);
- идентификатор узла (6 бит).

Каждая из четырех образованных подсетей может иметь до 62 узлов с идентификаторами от 1 до 62, идентификатор узла с номером 63 является широковещательным идентификатором для подсети.

Примечание. *Для идентификатора подсети можно выделять только старшие (самые левые) биты из части IP-адреса, отводимой под идентификатор узла.*

Примечание. *Возможность разбиения сетей на подсети обусловливается, в первую очередь, средствами маршрутизации IP-сегментов, а не средствами IP-модулей, формирующих и обрабатывающих IP-сегменты.*

Примечание. *Некоторые современные реализации протоколов маршрутизации для TCP/IP позволяют выделять "подподсети" в подсетях.*

3.3. Фрагментация IP-сегментов

Для того, чтобы существовала возможность передачи IP-сегментов через сети различного типа, межсетевой протокол обеспечивает адаптацию их размера к требованиям каждой сети. Это дает возможность, например, IP-сегментам, порожденным в сети на базе **Ethernet** (максимальный размер кадра - 1526 байт), беспрепятственно перемещаться до адресата по сети на базе **X.25** (максимальный размер кадра - 128 байт). Изменение размера IP-сегмента в процессе перемещения по сети может быть связано и с соображениями эффективности передачи.

Изменение размера IP-сегмента реализуется механизмом, называемым фрагментацией. IP-модуль на любом узле сети должен иметь возможность:

1. разбивать полученный им IP-сегмент на IP-фрагменты необходимого размера перед их передачей через конкретную сеть;
2. восстанавливать исходный IP-сегмент из получаемых им IP-фрагментов.

Каждый IP-фрагмент представляет собой полноценный IP-сегмент со своим собственным IP-заголовком. Однако заголовки всех IP-фрагментов содержат одинаковый идентификатор, совпадающий с идентификатором исходного IP-сегмента. Это позволяет распознавать все IP-фрагменты, относящиеся к одному исходному IP-сегменту.

IP-фрагменты в своих заголовках содержат поле "Смещение фрагмента", описывающее смещение данных IP-фрагмента в данных исходного IP-сегмента. Это поле позволяет корректно восстановить данные исходного IP-сегмента в принимающем IP-фрагменты узле даже в ситуации, когда IP-фрагменты приходят в порядке, от порядка их посылки (такое вполне возможно, т.к. IP-фрагменты могут следовать от источника к адресату по разным маршрутам).

Рассмотрим процесс фрагментации более подробно на следующем примере. IP-модуль на некотором узле получил IP-сегмент с идентификатором 9876 и данными длиной 300 байт (при этом бит запрета фрагментации DF установлен в 0). Этот IP-сегмент должен быть передан дальше к адресату через сеть, максимальный размер кадра которой равен 128 байтам.

Рис. 3.4 схематично представляет разбиение исходного IP-сегмента на 3 IP-фрагмента.

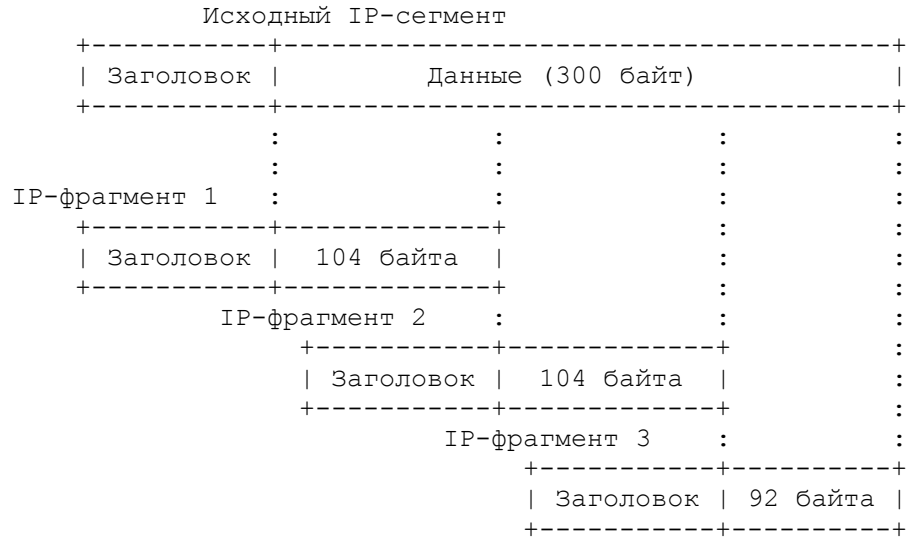


Рис. 3.4

IP-фрагмент 1 содержит в своем заголовке следующую информацию:

1. идентификатор - 9876;
2. длина заголовка - 5 (четырехбайтных слов);
3. длина сегмента - 124 (байт);
4. бит MF - 1;
5. смещение фрагмента - 0 (восьмибайтных единиц).

IP-фрагмент 2 содержит в своем заголовке следующую информацию:

1. идентификатор - 9876;

2. длина заголовка - 5 (четырехбайтных слов);
3. длина сегмента - 124 (байт);
4. бит MF - 1;
5. смещение фрагмента - 13 (восьмибайтных единиц).

IP-фрагмент 3 содержит в своем заголовке следующую информацию:

1. идентификатор - 9876;
2. длина заголовка - 5 (четырехбайтных слов);
3. длина сегмента - 112 (байт);
4. бит MF - 0;
5. смещение фрагмента - 26 (восьмибайтных единиц).

Заметим, что т.к. смещение фрагмента измеряется в восьмибайтных единицах, то длина данных в каждом IP-фрагменте (кроме последнего в цепочке) обязательно должна быть кратна 8. Вот почему в нашем примере это 104 байта (13 восьмибайтных единиц), а не 108, как допускает максимальная длина кадра в 128 байт (128 - 20 = 108, где 20 - длина заголовка).

IP-модуль на принимающем IP-фрагменты узле в ситуации, когда он должен транслировать IP-сегмент далее по сети, имеет три варианта действий с фрагментами:

1. переслать IP-фрагменты далее неизменными;
2. разбить (если в этом есть необходимость) полученные IP-фрагменты на более короткие IP-фрагменты;
3. восстановить исходный IP-сегмент из фрагментов.

В работе с IP-фрагментами на принимающей стороне используется специальный таймер, который с приходом первого фрагмента IP-сегмента устанавливается в исходное состояние (для UNIX-реализаций это, обычно, 30 сек) и начинает обратный счет. До момента обнуления таймера должны прийти все IP-фрагменты, относящиеся к этому сегменту. Если этого не произойдет, то все частично полученные данные IP-сегмента сбрасываются, а сам IP-сегмент считается утерянным.

3.4. Дополнительные данные IP-заголовка

Ниже кратко описываются дополнительные данные, которые могут включаться в IP-заголовок в случае необходимости.

Предписываемый маршрут

> список IP-адресов узлов сети, через которые должен следовать до адресата IP-сегмент. Предписываемый маршрут может быть "строгим" или "мягким". В первом случае IP-сегмент должен следовать строго только по указанным в списке узлам сети, во втором - допустимо прохождение через любые промежуточные узлы, не указанные в списке.

Пройденный маршрут

> список IP-адресов узлов сети, которые посетил IP-сегмент по пути к адресату. Каждый транзитный узел, через который следует IP-сегмент, помещает в этот список свой IP-адрес.

Временные метки (time stamp)

список моментов времени прохождения IP-сегмента через узлы сети, составляющие маршрут.

Секретность

указание на обработку IP-сегмента в соответствии с требованиями безопасности (RFC 1038). Эта возможность имеется только в нескольких (военных) реализациях TCP/IP.

Флаг окончания

указание на завершение дополнительных данных IP-заголовка.

Каждый элемент дополнительных данных представляет собой

- либо однобайтовый идентификатор дополнительных данных (например, "флаг окончания");
- либо комбинацию однобайтового идентификатора, поля длины и данных (например, "предписываемый маршрут").

Для дополнительных данных, пополняемых в ходе продвижения IP-сегмента по сети (например, "пройденный маршрут"), источник IP-сегмента должен зарезервировать место необходимого объема в IP-заголовке. Такой подход обеспечивает упрощение (а, следовательно, и ускорение) обработки IP-сегмента в узлах маршрута.

RFC 1063 (1988 г.) предлагает механизм определения оптимального размера IP-сегмента, при посылке его к определенному адресату. Этот механизм использует дополнительные данные IP-заголовка, называемые **probe MTU** (probe Maximum Transfer Unit - тестовый максимальный блок передачи). Каждый узел в маршруте IP-сегмента, содержащего такие дополнительные данные, сравнивает MTU следующей по маршруту сети с MTU, содержащемся в заголовке, и заменяет в нем старое значение на новое, если новое оказывается меньше. Конечный адресат IP-сегмента возвращает определенное таким образом значение источнику IP-сегмента. Использование в дальнейших посылках найденного размера IP-сегментов, позволяет избежать их фрагментации. Этот механизм широкого распространения еще не получил.

Примечание. *Понятие MTU подробно рассматривается в "Протоколы сетевого уровня"* .

4. Протокол управления передачей TCP

Протокол управления передачей TCP (Transmission Control Protocol) является протоколом транспортного уровня и базируется на возможностях, предоставляемых межсетевым протоколом IP. Основная задача TCP - обеспечение надежной передачи данных в сети. Его транспортный адрес в заголовке IP-сегмента равен 6. Описание протокола TCP дано в RFC 793.

Его основные характеристики перечислены ниже:

- реализует взаимодействие в режиме с установлением логического (виртуального) соединения;
- обеспечивает двунаправленную дуплексную связь;
- организует потоковый (с точки зрения пользователя) тип передачи данных;
- дает возможность пересылки части данных, как "экстренных";
- для идентификации партнеров по взаимодействию на транспортном уровне использует 16-битовые "номера портов";
- реализует принцип "скользящего окна" (sliding window) для повышения скорости передачи;
- поддерживает ряд механизмов для обеспечения надежной передачи данных.

Несмотря на то, что для пользователя передача данных с использованием протокола TCP выглядит как потоковая, на самом же деле обмен между партнерами осуществляется посредством пакетов данных, которые мы будем называть "TCP-пакетами".

4.1. Заголовок TCP-пакета

На рис. 4.1 приведен формат заголовка TCP-пакета.

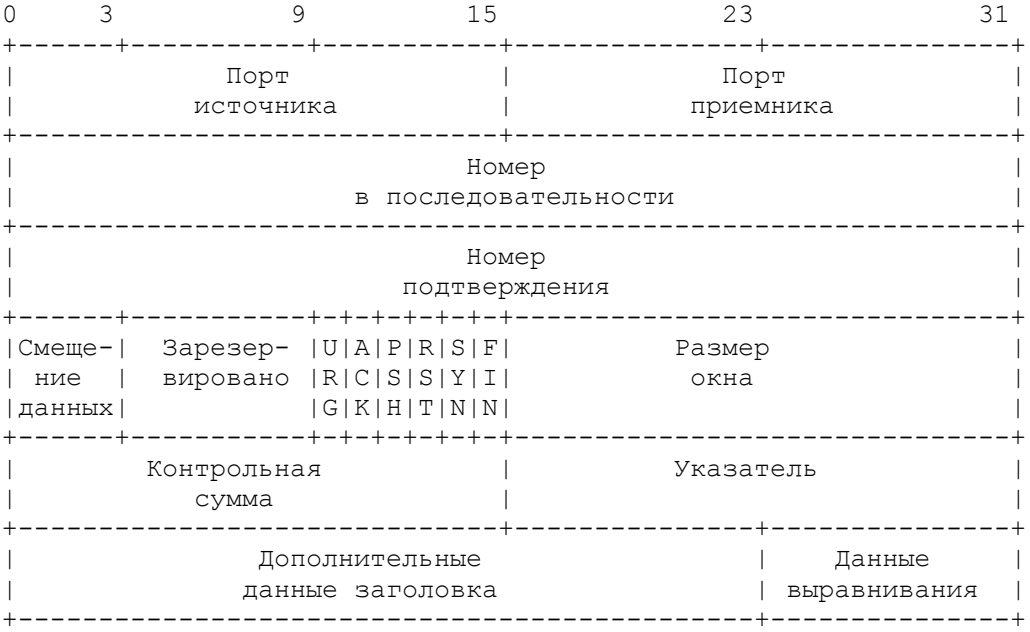


Рис. 4.1

Порт источника и порт приемника

16-битовые поля, содержащие номера портов, соответственно, источника и адресата TCP-пакета. Подробное описание понятия "номер порта" дано в "Номер порта".

Номер в последовательности (sequence number)

32-битовое поле, содержимое которого определяет (косвенно) положение данных TCP-пакета внутри исходящего потока данных, существующего в рамках текущего логического соединения.

В момент установления логического соединения каждый из двух партнеров генерирует свой начальный "номер в последовательности", основное требование к которому - не повторяться в промежутке времени, в течение которого TCP-пакет может находиться в сети (по сути, это время жизни IP-сегмента). Партнеры обмениваются этими начальными номерами и подтверждают их получение. Во время отправления TCP-пакетов с данными поле "номер в последовательности" содержит сумму начального номера и количества байт ранее переданных данных.

Номер подтверждения (acknowledgement number)

32-битовое поле, содержимое которого определяет (косвенно) количество принятых данных из входящего потока к TCP-модулю, формирующему TCP-пакет.

Смещение данных

четырехбитовое поле, содержащее длину заголовка TCP-пакета в 32-битовых словах и используемое для определения начала расположения данных в TCP-пакете.

Флаг URG

бит, установленное в 1 значение которого означает, что TCP-пакет содержит важные (urgent) данные. Подробно о данных этого типа сказано в "Важные данные".

Флаг ACK

бит, установленное в 1 значение которого означает, что TCP-пакет содержит в поле "номер подтверждения" верные данные.

Флаг PSH

бит, установленное в 1 значение которого означает, что данные содержащиеся в TCP-пакете должны быть немедленно переданы прикладной программе, для которой они адресованы. Подтверждение для TCP-пакета, содержащего единичное значение во флаге PSH, означает, что и все предыдущие TCP-пакеты достигли адресата.

Флаг RST

бит, устанавливаемый в 1 в TCP-пакете, отправляемом в ответ на получение неверного TCP-пакета. Также может означать запрос на переустановление логического соединения.

Флаг SYN

бит, установленное в 1 значение которого означает, что TCP-пакет представляет собой запрос на установление логического соединения. Получение пакета с установленым флагом SYN должно быть подтверждено принимающей стороной.

Флаг FIN

бит, установленное в 1 значение которого означает, что TCP-пакет представляет собой запрос на закрытие логического соединения и является признаком конца потока данных, передаваемых в этом направлении. Получение пакета с установленым флагом FIN должно быть подтверждено принимающей стороной.

Размер окна

16-битовое поле, содержащее количество байт информации, которое может принять в свои внутренние буфера TCP-модуль, отправляющий партнеру данный TCP-пакет. Данное поле используется принимающим поток данных TCP-модулем для управления интенсивностью этого потока: так, установив значение поля в 0, можно полностью остановить передачу данных, которая будет возобновлена только, когда размер окна примет достаточно большое значение. Максимальный размер окна зависит от реализации, в некоторых реализациях максимальный размер может устанавливаться системным администратором (типичное значение максимального размера окна - 4096 байт). Определение оптимального размера окна - одна из наиболее сложных задач реализации протокола TCP (см. "Исключение малых окон").

Контрольная сумма

16-битовое поле, содержащее Internet-контрольную сумму, подсчитанную для TCP-заголовка, данных пакета и псевдозаголовка. Псевдозаголовок включает в себя ряд полей IP-заголовка и имеет показанную на рис. 4.2 структуру.

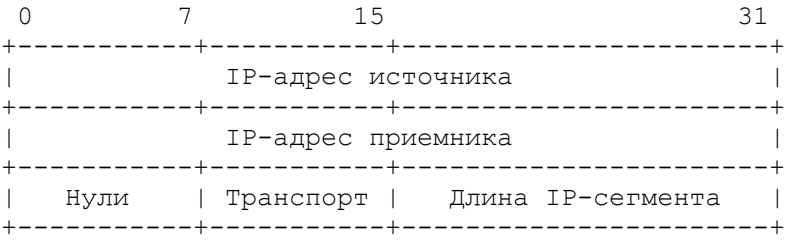


Рис. 4.2

Указатель

16-битовое поле, содержащее указатель (в виде смещения) на первый байт в теле TCP-пакета, начинающий последовательность важных (urgent) данных. Данные этого типа и механизм их обработки описаны в "Важные данные".

Дополнительные данные заголовка

последовательность полей произвольной длины, описывающих необязательные данные заголовка. Протокол TCP определяет только три типа дополнительных данных заголовка:

- конец списка полей дополнительных данных;
- пусто (No Operation);
- максимальный размер пакета.

Дополнительные данные последнего типа посылаются в TCP-заголовке в момент установления логического соединения для выражения готовности TCP-модулем

принимать пакеты длиннее 536 байтов. В UNIX-реализациях длина пакета обычно определяется максимальной длиной IP-сегмента для сети.

4.2. Номер порта

Номера портов играют роль адресов транспортного уровня, идентифицируя на конкретных узлах сети, по сути дела, потребителей транспортных услуг, предоставляемых как протоколом TCP, так и протоколом UDP. При этом протоколы TCP и UDP имеют свои собственные адресные пространства: например, порт номер 513 для TCP не идентичен порту номер 513 для UDP.

Примечание. *Своя собственная адресация на транспортном уровне стека протоколов сетевого взаимодействия необходима для обеспечения возможности функционирования на узле сети одновременно многих сетевых приложений. Наличие в TCP-заголовке номера порта позволяет TCP-модулю, получающему последовательности TCP-пакетов, формировать раздельные потоки данных к прикладным программам.*

Взаимодействие прикладных программ, использующих транспортные услуги протокола TCP (или UDP), строится согласно модели **"клиент-сервер"**, которая подразумевает, что одна программа (сервер) всегда пассивно ожидает обращения к ней другой программы (клиента). Связь программы-клиента и сервера идентифицируется пятеркой:

1. используемый транспортный протокол (TCP или UDP);
2. IP-адрес сервера;
3. номер порта сервера;
4. IP-адрес клиента;
5. номер порта клиента.

Для того, чтобы клиент мог обращаться к необходимому ему серверу, он должен знать номер порта, по которому сервер ожидает обращения к нему ("слушает сеть"). Для прикладных программ, получивших наибольшее распространение в сетях на основе TCP/IP, номера портов фиксированы и носят название **"хорошо известных номеров портов"** (well-known port numbers). В UNIX-системах такие номера портов содержатся в файле /etc/services. Ниже приводятся примеры хорошо известных номеров портов для некоторых серверов (служб).

```
Служба     Номер порта   Протокол
--------------------------------
ftp-data      20           TCP
ftp           21           TCP
telnet        23           TCP
smtp          25           TCP
time          37           TCP
time          37           UDP
finger        79           TCP
portmap      111           TCP
portmap      111           UDP
exec         512           TCP
login        513           TCP
shell        514           TCP
who          513           UDP
talk         517           UDP
route        520           UDP
Xserver     6000           TCP
```

Примечание. *Обратите внимание, что некоторые серверы (такие, например, как для службы portmap с номером порта 111) могут работать как по протоколу TCP, так и по протоколу UDP.*

Программы-клиенты, являющиеся активной стороной во взаимодействии "клиент-сервер", могут использовать, как правило, произвольные номера портов, назначаемые динамически непосредственно перед обращением к серверу (как любые свободные на данном узле).

Примечание. *Любая прикладная программа (будь то клиент или сервер) может открывать для взаимодействия любое количество портов для использования любых транспортных протоколов.*

Средства разработки сетевых приложений на базе транспортных протоколов TCP и UDP описаны в "Сетевое программирование".

4.3. Принцип "скользящего окна"

Протоколы транспортного уровня, обеспечивающие надежную передачу данных, предполагают обязательное подтверждение принимающей стороной правильности полученных данных.

В "простых" протоколах сторона, отправляющая данные, отсылает пакет с данными принимающей стороне и переходит в состояние ожидания подтверждения получения правильных данных. Только после приема подтверждения становится возможной следующая посылка. Очевидно, что такой подход использует пропускную способность сети неэффективно.

В протоколе TCP используется более совершенный принцип **"скользящего окна" (sliding window)**, который заключается в том, что каждая сторона может отправлять партнеру максимум столько байт, сколько партнер указал в поле "размер окна" заголовка TCP-пакета, подтверждающего получение предыдущих данных.

Принцип "скользящего окна" обеспечивает "опережающую" посылку данных с "отложенным" их подтверждением. Следует отметить недостаток этого механизма: если в течение некоторого времени не будет получено "отсроченное" подтверждение ранее отправленного пакета, то отправляющий TCP-модуль будет вынужден повторить посылку всех TCP-пакетов, начиная с неподтвержденного.

Размер окна, как правило, определяется объемом свободного места в буферах принимающего TCP-модуля.

4.4. Важные данные

Протокол TCP предусматривает возможность информирования принимающей стороны взаимодействия отправляющей стороной о наличии в TCP-пакете важных данных (**urgent data**), требующих особого внимания согласно логике прикладной задачи.

Примечание. *Отличие важных данных от данных основного потока заключается в том, что принимающая сторона должна, как правило, обработать их прежде ранее полученных, но еще не обработанных данных потока.*

Для индикации наличия в TCP-пакете важных данных используется флаг URG TCP-заголовка, местоположение важных данных в теле TCP-пакета определяется полем "Указатель" TCP-заголовка - оно задает смещение (в стиле языка программирования C) первого байта важных данных в теле TCP-пакета. Рис. 4.3 иллюстрирует расположение важных данных в теле TCP-пакета.

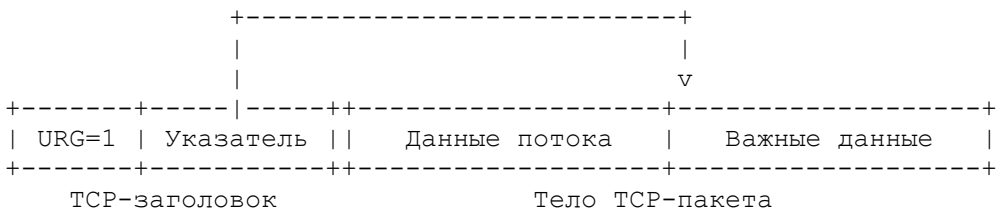


Рис. 4.3.

Примечание. *Протокол TCP предусматривает передачу важных (urgent) данных в рамках общего потока данных ("in-band"). Существуют протоколы (например, ISO), поддерживающие режим передачи важных (expedited) данных вне общего потока данных ("out-band"), что в общем случае быстрее.*

4.5. Этапы TCP-взаимодействия

Взаимодействие партнеров с использованием протокола TCP строится в три этапа:

- установление логического соединения;
- обмен данными;
- закрытие соединения.

Ниже с помощью трех рисунков дается описание каждого из этапов. Рисунки иллюстрируют последовательность обмена TCP-пакетами двумя TCP-модулями: A и B. TCP-пакеты представлены тремя полями TCP-заголовка ("Номер в последовательности", "Номер подтверждения", "Флаги") и числом, характеризующим длину данных, составляющих тело TCP-пакета (заметим, что реально поля длины данных в TCP-заголовке нет). Стрелками показаны направления пересылки пакетов.

TCP A	Номер в последовательности	Номер подтверждения	Флаги	Длина данных	TCP B
--->	1000		SYN	0	
	5000	1001	SYN,ACK	0	<---
--->	1001	5001	ACK	0	

Рис. 4.4

Рис. 4.4 иллюстрирует этап установления соединения, реализуемый как **"трехшаговое рукопожатие"** (three-way handshake). На первом шаге TCP-модуль A, играя роль клиента, посылает TCP-модулю B пакет с установленным флагом SYN и начальным значением номера в последовательности равным 1000. TCP-модуль B, будучи готов со своей стороны установить соединение, отвечает TCP-пакетом, подтверждающим правильный прием запроса (поле "Номер подтверждения" на 1 больше начального номера в

последовательности для TCP-модуля A и среди флагов есть установленный в 1 флаг ACK) и информирующим о готовности установить соединение (взведен флаг SYN и установлен в 5000 начальный номер в последовательности). На третьем шаге TCP-модуль A подтверждает правильность приема TCP-пакета от B.

Примечание. *Некоторые протоколы транспортного уровня (но не TCP) допускают обмен данными уже на этапе установления логического соединения.*

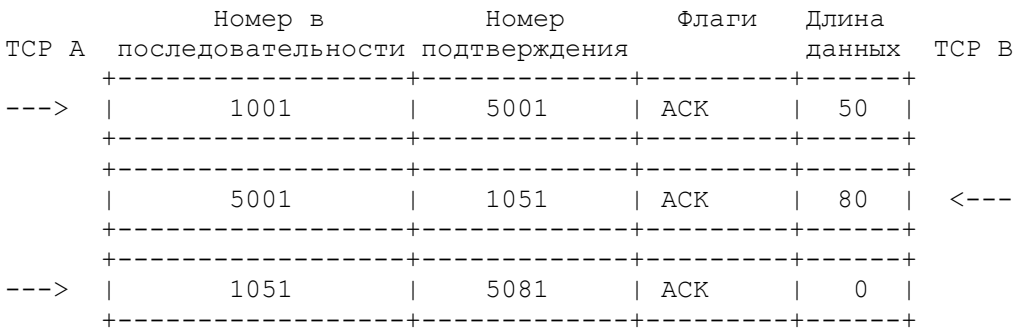


Рис. 4.5

Рис. 4.5 иллюстрирует этап двустороннего обмена данными между TCP-модулями A и B. TCP-модуль, принимающий адресованные ему данные, всегда подтверждает их прием, вычисляя значение поля "Номер подтверждения" в заголовке ответного TCP-пакета как сумму пришедшего "Номера в последовательности" и длины правильно принятых данных. Отметим, что посылка данных к партнеру и подтверждение принятых от него данных реализуются в рамках одного TCP-пакета.

Рис. 4.6

Рис. 4.6 иллюстрирует закрытие соединения по инициативе TCP-модуля A, посылающего партнеру TCP-пакет с установленным флагом FIN. Прием запроса на закрытие соединения TCP-модуль B подтверждает пакетом, содержащем в своем заголовке поле "Номер подтверждения", значение которого (1052) на 1 больше значения принятого "Номера в последовательности" (1051). После этого посылка каких-либо данных TCP-модулем A становится невозможной, однако модуль B имеет данные для передачи, которые он

отправляет TCP-модулю A и получает подтверждение на их прием. Затем TCP-модуль B формирует пакет с флагом FIN, после подтверждения его приема соединение считается закрытым.

Примечание. *Обратите внимание на то обстоятельство, что при подтверждении TCP-пакетов, содержащих единичные флаги SYN или FIN, значение поля "Номер подтверждения" на 1 больше значения соответствующего поля "Номер в последовательности", несмотря на то, что никакие данные в подтверждаемых TCP-пакетах не передаются.*

Примечание. *Рассмотренный пример не включает в себя ситуации, связанные с "потерей" TCP-пакетов в сети, и их обработку, связанную с повторной передачей данных.*

4.6. Таймеры

4.6.1. Таймер повторной передачи

Данный таймер взводится значением **RTO** (**Retransmission TimeOut** - интервал до повторной передачи) в момент посылки TCP-пакета адресату. Если таймер окажется сброшенным в ноль до момента получения подтверждения пакета, то этот пакет должен быть послан вновь.

Ясно, что величина RTO не может быть фиксированной, т.к. TCP-пакеты до разных адресатов следуют по различным маршрутам через сети, скорость передачи данных в которых может различаться более чем в тысячи раз. Для вычисления "оптимального" значения RTO в каждом логическом соединении используется специальная процедура, специфицированная в RFC 793.

Согласно этой процедуре, для каждого TCP-пакета измеряется величина **RTT** (**Round Trip Time** - интервал времени от момента посылки TCP-пакета до момента получения подтверждения на него). На основе измеренных RTT вычисляется величина **SRTT** (**Smoothed RTT** - сглаженный RTT) по следующей формуле:

```
SRTT = k*SRTT + (1 - k)*RTT,
```

где **k** - сглаживающий коэффициент (например, 0.9).

Примечание. *Приведенная формула обеспечивает фильтрацию нетипичных (пиковых) значений измеренной величины RTT.*

"Оптимальное" значение RTO вычисляется по формуле:

```
RTO = min(U, max(L, p*SRTT)),
```

где:

U - ограничение сверху на значение RTO (например, 30 секунд);

L - ограничение снизу на значение RTO (например, 1 секунда);

p - коэффициент "запаса" (например, 2).

Если после повторной посылки TCP-пакета, опять не будет получено его подтверждение за интервал времени RTO, то попытки послать TCP-пакеты будут повторены (до 12 раз), но каждый раз с экспоненциально возрастающим значением RTO. Только после неудачи всей серии повторных посылок связь между партнерами будет считаться аварийно закрытой.

4.6.2. Таймер возобновления передачи

В ходе взаимодействия двух TCP-модулей (**A** и **B**) вполне возможна следующая ситуация:

- TCP-модуль B уведомляет TCP-модуль A о невозможности приема от него данных, определяя размер окна равным 0;
- TCP-модуль A, имея данные для передачи, переходит в состояние ожидания от TCP-модуля B пакета с ненулевым размером окна;
- TCP-модуль B, у которого освободилось некоторое пространство в буферах, посылает модулю A TCP-пакет с ненулевым размером окна;
- адресованный модулю A пакет "теряется" по какой-либо причине и оба TCP-модуля переходят в состояние бесконечного ожидания.

Средством выхода из такого тупикового состояния и служит таймер возобновления **передачи** (**persistence timer** - "настойчивый" таймер). Он взводится в момент получения TCP-пакета с нулевым значением поля "Размер окна" в его заголовке (типичное начальное значение для этого таймера - 5 секунд). Если до момента обнуления таймера не будет получено разрешение на возобновление передачи данных, то ожидающий разрешения TCP-модуль отправляет партнеру пакет, содержащий всего лишь 1 байт данных. По реакции партнера, возвращающего пакет с нулевым/ненулевым значением размера окна, TCP-модуль продолжает ожидание или возобновляет посылку данных.

4.6.3. Таймер закрытия связи

Протокол TCP предусматривает следующий простой прием предотвращения появления в сети TCP-пакетов, не имеющих адресатов: после закрытия логического соединения между партнерами номера портов, использовавшихся в этом соединении, остаются еще некоторый интервал времени действительными, что дает возможность долго блуждавшим по сети TCP-пакетам добраться до места назначения (где они будут просто проигнорированы). Величина этого интервала равна удвоенному времени жизни IP-сегмента (обычно, 2*15=30 секунд).

Примечание. *Пользователи ОС UNIX могут почувствовать эффект от использования этого приема, попытавшись перезапустить некоторую прикладную программу, использующую TCP, сразу же после ее завершения.*

4.6.4. Таймеры поддержки соединения

Ниже описывается механизм, используемый для проверки ненарушенности логического соединения между TCP-модулями.

Каждый TCP-модуль, участвующий в логическом соединении, через фиксированный промежуток времени (**keep-alive timer**), равный обычно 45 секундам, периодически отправляет партнеру пустые (не содержащие данных) TCP-пакеты и ждет их подтверждения. Каждое полученное подтверждение говорит о ненарушенности соединения. Если же в течении определенного интервала времени (**idle timer**), равного

обычно 360 секудам, не будет получено ни одного подтверждения, то логическое соединение считается оборванным.

Примечание. *Очевидно, что данный механизм имеет смысл включать в работу только тогда, когда партнеры по TCP-взаимодействию приостановили по какой-либо причине обмен данных на достаточно длительный срок (более 45 секунд).*

Примечание. *Стандартная спецификация протокола TCP не включает в себя описанный механизм, однако он реализован во всех UNIX-системах.*

4.7. Алгоритмы повышения эффективности

Ниже описываются некоторые алгоритмы, используемые для повышения эффективности взаимодействия по протоколу TCP в UNIX-реализациях и не являющиеся частью спецификации TCP.

4.7.1. Задержка подтверждения

Задержка отсылки подтверждения принятого пакета используется для сокращения числа TCP-пакетов, которыми обмениваются партнеры по взаимодействию. Поясним эффект от такой задержки следующим примером.

Пусть клиентская часть некоторого приложения (например, службы telnet) направляет серверной части некоторые данные (в случае telnet - строку символов, представляющих команду ОС UNIX, которая должна быть выполнена на удаленном узле сети). Серверная часть, получив данные и обработав их, должна вернуть клиенту результат (в случае с telnet - это стандартный вывод исполненной команды).

В ситуации без задержки TCP-модуль на стороне сервера, приняв пакет с данными и разместив их в своем буфере, сразу же отвечает подтверждающим пакетом, содержащим в своем заголовке и некоторый (уменьшенный) размер окна для приема последующих данных. Спустя некоторое (обычно, очень короткое) время данные из буфера передаются серверной части прикладной программы. Освобождение места в буфере заставляет TCP-модуль отправлять партнеру на стороне клиента TCP-пакет с новым (увеличившимся) размером окна. Тем временем прикладная программа, обработав полученные данные (часто за небольшое время), передает результат TCP-модулю для отсылки его клиенту, для чего модуль формирует еще один пакет. Итого: одна транзакция потребовала от TCP-модуля на стороне сервера посылки трех TCP-пакетов.

Введение же задержки при отсылке подтверждающего TCP-пакета позволяет в ряде случаев уменьшить количество пакетов с трех до одного, содержащего сразу подтверждение, новый размер окна и результирующие данные. Экспериментальные исследования показали, что во многих случаях "оптимальным" значением задержки является 0.2 секунды.

Для того, чтобы введение задержки сказывалось минимальным образом на приложения, предъявляющие жесткие требования к пропускной способности сети, задержка устанавливается нулевой при условии, что размер окна изменяется более чем на 35% или (в абсолютном исчислении) на удвоенный максимальный размер TCP-пакета.

4.7.2. Исключение малых окон

Возможны ситуации, когда прикладная программа, использующая TCP-сервис, "выбирает" из буфера обслуживающего ее TCP-модуля пришедшие для нее данные малыми порциями. Это приводит к генерации TCP-модулем большого количества TCP-пакетов, содержащих в своих заголовках малую величину размера окна, что в свою очередь приводит к генерации на передающей данные стороне многих TCP-пакетов с "короткими" данными. Как результат - "засорение" сети короткими пакетами и снижение ее пропускной способности.

Во избежание деградации сети вследствие описанного явления используется следующий прием: TCP-пакет, информирующий посылающую данные сторону об увеличении размера окна, формируется только при выполнении одного из двух условий:

1. свободное место в буфере принимающего данные TCP-модуля увеличилось по крайней мере на четверть размера этого буфера;
2. свободное место увеличилось по крайней мере на максимальный размер TCP-пакета.

Кроме того TCP-модуль, отправляющий данные, должен делать это большими порциями.

4.7.3. Исключение коротких TCP-пакетов

"Засорение" сети короткими TCP-пакетами возможно и в ситуации, когда прикладная программа, отправляющая данные партнеру по взаимодействию, делает это короткими порциями (типичный пример - любая программа, использующая графическую систему X Window System).

Для борьбы с этим используется следующий прием:

- самая первая порция данных отправляется TCP-модулем сразу же при поступлении "коротким" TCP-пакетом;
- все последующие накапливаются в буфере TCP-модуля, пока их общий объем не составит максимального размера TCP-пакета или не будет получено подтверждение предеыдущей посылки.

Однако этот подход может сказаться на быстродействии некоторых приложений, чтобы избежать этого прикладной программе предоставляются средства для принудительного "выталкивания" буферизованных данных в необходимых случаях. Кроме того, существует возможность отключения описанного механизма.

4.7.4. Алгоритм медленного старта

Опыт эксплуатации сетей на основе TCP/IP показал, что с повышением загрузки сети (особенно, сети со шлюзом) ее пропускная способность падает (хотя, казалось бы, она должна оставаться постоянной). Исследования показали, что падение обусловлено появлением в сети большого числа TCP-пакетов, повторно посылаемых к активно используемому узлу сети (обычно это шлюз в другие сети). Дело в том, что приемный буфер TCP-модуля на шлюзе очень быстро заполняется, и TCP-модуль вынужден сбрасывать поступающие к нему пакеты.

Для предупреждения подобной ситуации необходимо согласование темпа передачи TCP-пакетов с возможностями их приема на узле-адресате. Задачу согласования решает алгоритм медленного старта, постепенно повышающий темп передачи данных от

медленного до "оптимального", при котором нет повторных передач TCP-пакетов. Алгоритм использует так называемое "**окно перегруженности**" (**congestion window**), используемое на передающей стороне для определения максимального объема передаваемых данных вместо размера, получаемого от принимающей стороны в поле окна подтверждающего пакета.

Размер "окна перегруженности" определяется на передающей стороне путем постепенного его увеличения до момента появления повторных передач (ясно, что размер этого окна никогда не превышает размера окна на принимающей стороне). Однажды определенный размер "окна перегруженности" остается неизменным, пока вновь не появятся повторные передачи, однако периодически делаются осторожные попытки и увеличить этот размер.

Эксперименты показали, что данный алгоритм позволяет уменьшить количество повторно передаваемых TCP-пакетов на 50% и повысить пропускную способность сети на 30%.

5. Протокол дэйтаграмм пользователя UDP

Протокол дэйтаграмм пользователя UDP (User Datagram Protocol) является протоколом транспортного уровня и базируется на возможностях, предоставляемых межсетевым протоколом IP. Основная задача TCP - обеспечение "быстрой" передачи данных в сети. Его транспортный адрес в заголовке IP-сегмента равен 17. Описание протокола UDP дано в RFC 768.

Его основные характеристики перечислены ниже:

- реализует взаимодействие в режиме без установлением логического (виртуального) соединения;
- организует поблочный (дэйтаграммный, пакетный) тип передачи данных;
- для идентификации партнеров по взаимодействию на транспортном уровне использует 16-битовые "номера портов";
- не гарантирует надежной передачи данных (возможна как потеря UDP-пакетов, так и их дублирование);
- не имеет средств уведомления источника UDP-пакета о правильности/ошибочности в его приеме адресатом;
- не обеспечивает правильный порядок доставки UDP-пакетов от источника к приемнику;
- может гарантировать целостность данных в UDP-пакете за счет использования контрольной суммы;
- очень прост (особенно, по сравнению с протоколом TCP).

Следует отметить, что, по сути дела, протокол транспортного уровня UDP играет роль интерфейса для прикладных программ к средствам протокола межсетевого уровня IP.

На рис. 5.1 приведен формат заголовка UDP-пакета.

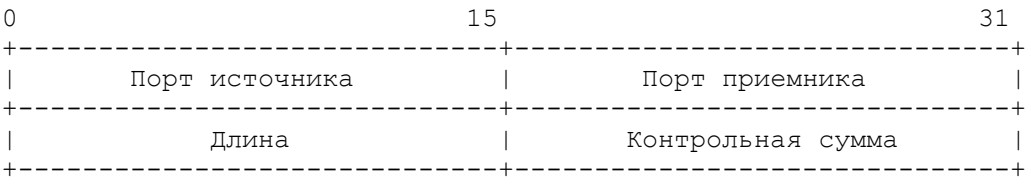


Рис. 5.1

Порт источника и порт приемника

16-битовые поля, содержащие номера портов, соответственно, источника и адресата UDP-пакета. Понятие "номер порта" обсуждается в "Протокол управления передачей TCP".

Длина

16-битовое поле, содержащее длину (в байтах) всего UDP-пакета, включая заголовок и данные.

Контрольная сумма

16-битовое поле, содержащее Internet-контрольную сумму, подсчитанную для UDP-заголовка, данных пакета и псевдозаголовка. Псевдозаголовок (такой же, как для подсчета контрольной суммы в TCP-заголовке) включает в себя ряд полей IP-заголовка и имеет показанную на рис. 5.2 структуру.

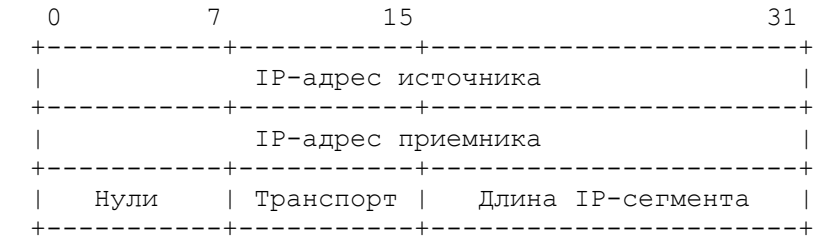


Рис. 5.2

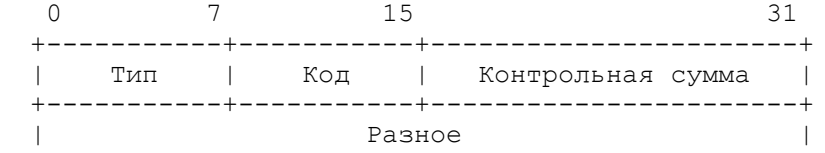
Если поле "Контрольная сумма" UDP-заголовка содержит нулевое значение, это означает, что источник UDP-пакета контрольную сумму не подсчитывал, и приемник выполнять ее проверку не должен. Некоторые реализации протокола UDP (например, в SunOS - клоне ОС UNIX от Sun Microsystems) контрольную сумму не подсчитывают в принципе, полагаясь на возможности контроля целостности данных, реализованные в протоколах сетевого уровня (например, в Ethernet).

6. Межсетевой протокол управляющих сообщений ICMP

Межсетевой протокол управляющих сообщений ICMP (Internet Control Message Protocol), специфицированный в RFC 792, играет роль транспортного протокола для управляющей и диагностической информации, которой обмениваются между собой IP-, TCP- или UDP-модули скрытно от приложений. Протокол ICMP поддерживается в обязательном порядке ка ждым IP-модулем. Его транспортный адрес в IP-заголовке равен 1.

6.1. Заголовок ICMP-пакета

Поскольку протокол ICMP используется для транспортировки весьма различной информации, то фиксируется лишь общая структура заголовка ICMP-пакета, имеющего формат, показанный на рис. 6.1.



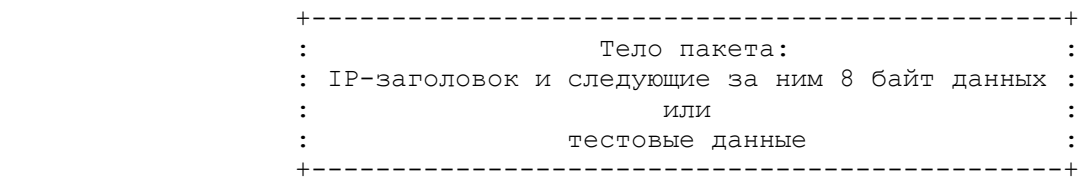


Рис. 6.1

Тип

однобайтовое поле, содержащее идентификатор типа ICMP-пакета. Возможные значения этого поля приведены в таблице.

Поле "Тип"	Назначение
0	Ответ на запрос эха
3	Адресат недоступен
4	Подавление источника
5	Перенаправление
8	Запрос эха
11	Исчерпано время жизни
12	Ошибка в параметре
13	Запрос временной метки
14	Ответ на запрос временной метки

Код

однобайтовое поле, значение которого конкретизирует назначение ICMP-пакета определенного типа.

Контрольная сумма

16-битовое поле, содержащее Internet-контрольную сумму, подсчитанную для всего ICMP-пакета целиком.

Разное

четырезбайтовое поле, предназначенное для хранения разнообразной информации, специфичной для ICMP-пакетов определенного типа (например, номера в TCP-последовательности, IP-адреса и т.п.).

Тело пакета

Здесь содержится заголовок IP-сегмента, явившегося порождения данного ICMP-пакета, и первые 8 байт данных тела этого IP-сегмента. Если ICMP-пакет есть результат проявления аномалии в TCP- или UDP-взаимодействии, то эти 8 байт будут представлять собой первые восемь байтов, соответственно, TCP- или UDP-заголовка, что дает возможность определить, в частности, номера портов (а, следовательно, и использующие их прикладные программы).

Для ICMP-пакетов некоторых типов это может содержать не начало IP-сегмента, а тестовые данные.

Источниками и обработчиками ICMP-пакетов могуть быть как IP-модули, так и TCP- и UDP-модули (но никогда прикладные программы).

Проблемы в доставке и обработке ICMP-пакетов никогда не приводят к порождению новых ICMP-пакетов, уведомляющих об этих проблемах. Сделано это с целью избежать возможных бесконечных циклов генерации ICMP-пакетов в сети.

6.2. Типы ICMP-пакетов

Здесь рассматриваются 6 типов ICMP-пакетов, реализованных во всех клонах и версиях ОС UNIX.

6.2.1. Адресат недоступен

ICMP-пакет этого типа генерируется в следующих случаях:

1. сеть, узел сети, протокол или порт являются недоступными;
2. в ходе продвижения по сети IP-сегмента потребовалась его фрагментация, однако в заголовке сегмента установлен флаг DF, запрещающий делать это;
3. предписываемый маршрут, указанный в поле дополнительных данных IP-сегмента, оказался недействительным (несуществующим или неактивным).

6.2.2. Подавление источника

В ситуациях, когда некоторый узел (как правило, шлюз) не имеет достаточно места в своих буферах для размещения интенсивно поступающих к нему данных, он может послать узлам-источникам ICMP-пакет данного типа (**source quench**). Узел-источник в ответ на такое уведомление обязан уменьшить темп передачи данных.

В ранних UNIX-реализациях протоколов TCP/IP ICMP-пакеты этого типа игнорировались. В TCP-реализациях, поддерживающих алгоритм медленного старта, в ответ на это сообщение уменьшается размер "окна перегруженности". UDP-модули игнорируют это сообщение, информируя при этом обслуживаемую прикладную программу о требовании приемника уменьшить интенсивность и/или размер дэйтаграмм.

6.2.3. Перенаправление

ICMP-пакет этого типа посылается источнику данных, когда узел-шлюз обнаруживает, что источник может направлять свои данные непосредственно к следующему шлюзу маршрута. Такой ICMP-пакет содержит в себе IP-адрес этого шлюза. Этот IP-адрес должен быть включен в таблицу маршрутизации на узле-источнике данных.

6.2.4. Эхо

Для реализации эха IP-модуль на узле A отправляет узлу B ICMP-пакет типа "запрос эха", содержащий в своем теле вместо IP-заголовка тестовые данные произвольной длины. Узел B, получив такой запрос, возвращает узлу A ICMP-пакет типа "ответ на запрос эха", содержащий те же данные, что и в запросе. Эхо-посылки используются для проверки достижимости удаленных узлов сети и измерения времени прохождения данных.

6.2.5. Исчерпано время жизни

ICMP-пакет данного типа посылается источнику IP-сегмента, который должен быть сброшен по одной из двух причин:

1) исчерпано время жизни IP-сегмента;

2) исчерпано допустимое время на сборку фрагментированного IP-сегмента.

6.2.6. Неверный параметр

С помощью ICMP-пакета данного типа источник IP-сегмента информируется о том, что данный сегмент сброшен вследствие наличия ошибки в каком-либо из полей его заголовка.

7. Протоколы сетевого уровня

Ниже кратко описывается реализация стека протоколов TCP/IP на базе ряда протоколов сетевого уровня.

7.1. Ethernet

Протокол Ethernet был разработан в начале 1970-х годов совместно фирмами **Xerox, DEC** и **Intel.** На его базе в 1982 г. был принят международный стандарт **IEEE 802.3** .

Использование протокола сетевого уровня Ethernet совместно с протоколами TCP/IP регламентируется RFC 894.

Основными характеристиками протокола Ethernet являются следующие:

- шинная логическая топология сети;
- скорость передачи данных 10 мегабит в секунду;
- используется для построения локальных вычислительных сетей;
- обмен данными между узлами сети осуществляется кадрами;
- для разделения шины между многими узлами используется механизм CSMA/CD;
- обеспечивает широковещательную (broadcast) и многопунктовую (multicast) рассылку данных.

В качестве физической среды передачи данных Ethernet использует:

- "толстый" коаксиальный кабель (так называемый 10base5 Ethernet);
- "тонкий" коаксиальный кабель (10base2);
- оптоволоконный кабель;
- витая пара (10baseT).

В первых трех случаях физическая топология сети реально является шинной, в последнем - физическая топология сети представляет собой "звезду".

Примечание. *Существуют современные версии Ethernet, обеспечивающие скорость передачи в 100 мегабит в секунду.*

Примечание. *Ethernet позволяет объединить в локальную сеть узлы, расположенные друг от друга на расстоянии от нескольких десятков метров (10baseT) до нескольких километров (сегменты 10base5, связанные повторителями).*

Механизм **CSMA/CD** (Carrier Sense Multiple Acces with Collision Detection - Множественный Доступ с Контролем Носителя и Обнаружением Столкновений) подразумевает следующий алгоритм получения узлом сети доступа к шине:

1. прослушивание шины (sense carrier) на предмет наличия в ней сигналов передачи данных другими узлами;
2. если шина занята, то отложить передачу, если свободна - начать передачу данных;
3. в течение первых 47 микросекунд передачи кадра данных вести проверку столкновений (collisions) в шине, связанных с возможным одновременным началом передачи данных и другими узлами сети;
4. при обнаружении столкновения прекратить передачу данных и перейти в состояние ожидания на период времени случайной длины, а потом возобновить попытки передачи кадра.

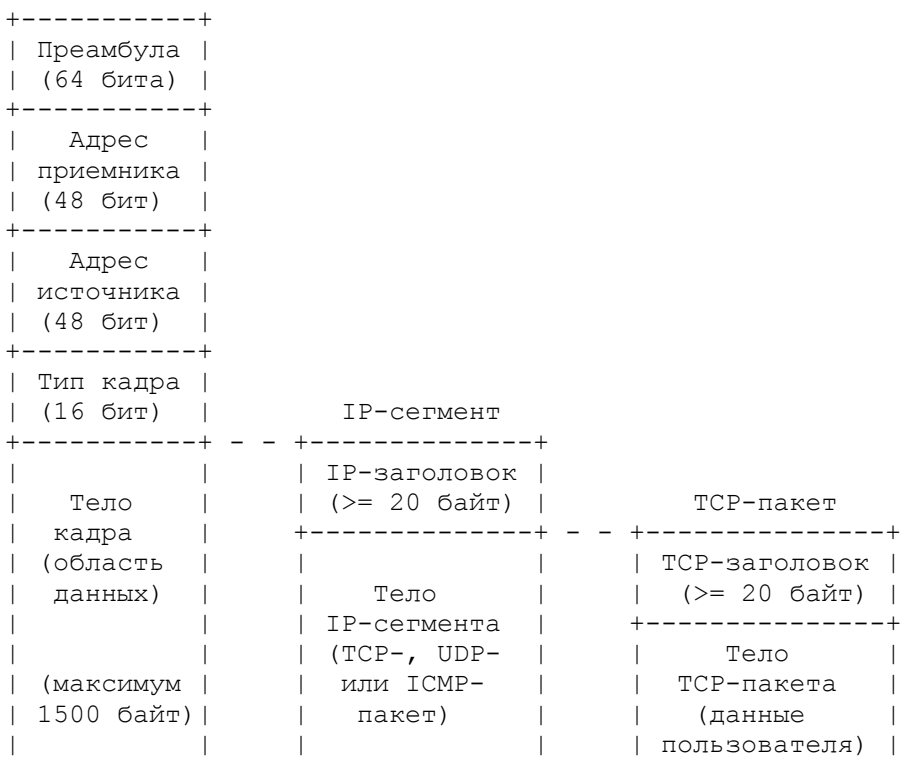
Обмен данными по протоколу Ethernet всегда реализуется программно-аппаратно с помощью двух компонентов:

- сетевого контроллера (чаще всего имеющего вид печатной платы, вставляемой в корпус ЭВМ), подключаемого к шине (коаксиальному кабелю, оптоволокну или витой паре медных проводов);
- драйвера сетевого контроллера, обеспечивающего интерфейс сетевого программного обеспечения (например, IP-модуля) с контроллером.

Примечание. *В ОС UNIX сетевой контроллер и его драйвер принято называть* ***"сетевым интерфейсом"****.*

7.1.1. Формат кадра данных Ethernet

На рис. 7.1 представлен формат кадра данных протокола Ethernet. Для иллюстративных целей показана вложенность в кадр IP-сегмента (содержащего, в свою очередь, TCP-пакет).



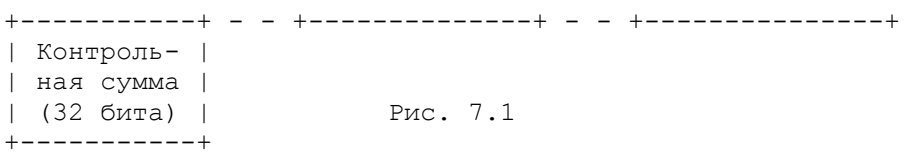


Рис. 7.1

Преамбула

64-битовое поле, содержащее фиксированную последовательность битов, используемую для синхронизации схем приема сигналов на узле-адресате.

Адрес приемника и адрес источника

48-битовые поля, содержащие Ethernet-адреса принимающего и передающего кадр узлов сети.

Каждый Ethernet-контроллер в мире имеет уникальный 6-байтовый адрес. Ethernet-адрес принято записывать в виде последовательности шести разделенных символом "двоеточие" двузначных шестнадцатиричных чисел, где каждое число представляет собой значение одного байта адреса, напрмер, f1:e2:d3:c4:b5:a6.

Примечание. *Как правило, Ethernet-адрес жестко "зашит" в контроллере, однако существуют контроллеры, допускающие его изменение программным путем.*

Тип кадра

16-битовое поле, содержащее идентификатор протокола вышележащего уровня, использующего данный Ethernet-кадр. Т.е. поле определяет принадлежность содержимого тела кадра. Наличие данного поля в кадре обеспечивает возможность функционирования в одной сети на базе Ethernet одновременного нескольких различных стеков протоколов, а не только одного TCP/IP.

Примерами значений данного поля являются следующие:

- 0x0800 - протокол IP;
- 0x0806 - протокол ARP.

Тело кадра

содержит данные, передаваемые в кадре протоколом вышележащего уровня (на рисунке это IP-сегмент, тело которого используется для пересылки TCP-пакета).

Максимальная длина тела кадра протокола сетевого уровня обозначается как **MTU (Maximum Transmission Unit)** и для Ethernet составляет 1500 байтов.

Контрольная сумма

32-битовое поле, содержащее CRC-контрольную сумму, подсчитанную для всего кадра.

Минимальная длина Ethernet-кадра составляет 64 байта (512 бит). Такое ограничение связано с тем, что контроль столкновений различных кадров в Ethernet-шине согласно алгоритму CSMA/CD выполняется на интервале времени в 47 микросекунд. За это время

осуществляется передача 470 бит (при скорости 10 мегабит в секунду), так что 512 - это округление 470 до числа, являющегося степенью 2.

В ситуациях, когда длина данных, передаваемых в теле кадра, недостаточна для формирования кадра длиной не менее 64 байтов, драйвер Ethernet-контроллера искусственно дополняет тело пакета до необходимой длины.

Примечание. *Интересно, что согласно стандарту IEEE 802.3 рассмотренное выше 16-битовое поле типа кадра на самом деле является полем длины (в байтах) тела Ethernet-кадра. Для идентификации типа содержимого тела кадра предлагается использовать специальный протокол LLC (Logic Link Control - протокол управления логической связью), занимающий промежуточное положение между Ethernet и вышележащими протоколами. Однако протокол LLC в среде UNIX (а, значит, и в большинстве других ОС) реализован не был: стандартом "de facto" остаются спецификации RFC 894. Хотя надо отметить, что выбор значений идентификаторов типа кадра (0x0800 и больше) не исключает возможности использования этого поля одновременно и для идентификации типа, и для хранения длины тела кадра (максимум 1500).*

7.1.2. Протоколы трансляции адресов

Ethernet подобно другим протоколам сетевого уровня обладает собственной системой адресации узлов сети, отличной от системы адресации, принятой в TCP/IP. Это приводит к необходимости взаимной трансляции адресов "IP-адрес в Ethernet-адрес" и обратно.

В UNIX-системах такая трансляция выполняется с помощью специальной таблицы соответствий пар адресов различного типа, которая динамически создается и обновляется сетевым интерфейсом. В момент активизации сетевого интерфейса содержимое таблицы трансляции может загружаться из созданного вручную специального административного файла, однако это не является обязательным.

Для поддержания таблицы трансляции в актуальном состоянии, отражающем текущий состав узлов Ethernet-сети, используется протокол **ARP (Address Resolution Protocol)**, описанный в RFC 826.

Структура ARP-сегмента приведена на рис. 7.2.

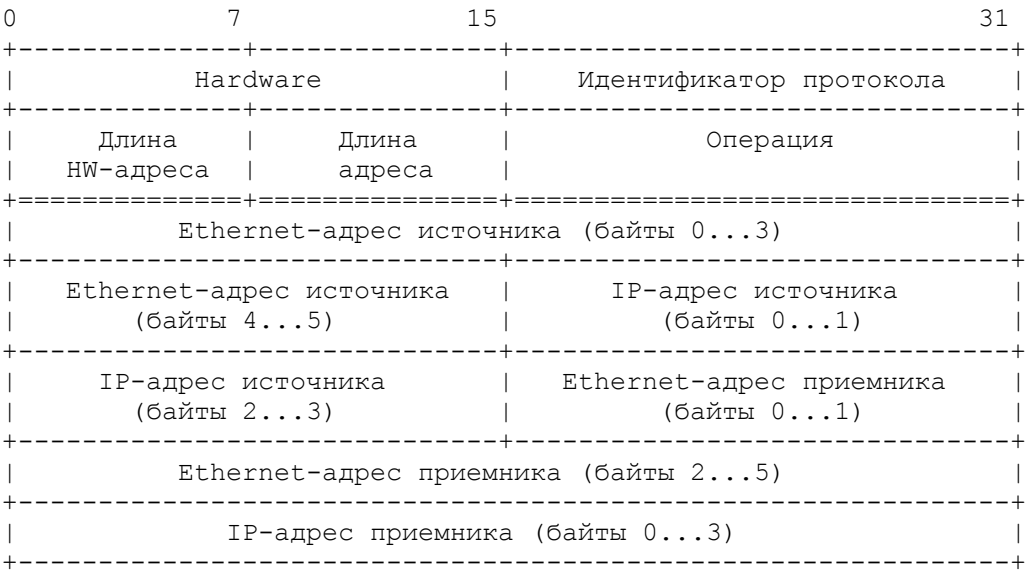


Рис. 7.2

Поле **"Hardware"** содержит идентификатор типа адреса на сетевом уровне (в нашем случае - Ethernet).

Поле **"Идентификатор протокола"** определяет протокол межсетевого уровня (в нашем случае - IP).

Поля длин задают длину адресов (в нашем случае: **"Длина HW-адреса"** равна 6, а **"Длина адреса"** равна 4).

Поле **"Операция"** содержит идентификатор типа ARP-сегмента (запрос или ответ).

Примечание. *Как видно из структуры ARP-сегмента протокол ARP может быть использован для совместной работы TCP/IP не только с протоколом Ethernet, но и с другими протоколами сетевого уровня, когда в этом есть необходимость.*

Алгоритм использования протокола ARP для построения таблицы трансляции на некотором узле сети (назовем его А) выглядит следующим образом.

1. На узле А IP-модуль передает сетевому интерфейсу сегмент для его пересылки узлу Б (в сегменте присутствует IP-адрес узла Б). Сетевой интерфейс просматривает свою таблицу трансляции адресов, пытаясь по известному IP-адресу узла Б определить его Ethernet-адрес. Если необходимая строка в таблице есть, то сетевой интерфейс формирует Ethernet-кадр и передает его в сеть.
2. Если нужной строки в таблице нет, то сетевой интерфейс строит ARP-сегмент, содержащий IP-адрес узла Б, упаковывает его в Ethernet-кадр. Этот кадр в качестве Ethernet-адреса приемника содержит широковещательный адрес, что обеспечит получение этого кадра всеми узлами локальной сети.
3. Все узлы локальной сети получают данный Ethernet-кадр, а в нем ARP-сегмент. IP-адрес из ARP-сегмента сравнивается с собственным IP-адресом и, если они совпадают (это должно иметь место только на узле Б), то собственный Ethernet-адрес возвращается узлу А в ответном ARP-сегменте.
4. Получив ответный ARP-сегмент, сетевой интерфейс на узле А добавляет в таблицу трансляции новую строку, содержимое которой и будет использовано для посылки IP-сегмента к узлу Б.

Для того, чтобы таблица трансляции адресов с малым временем реакции отслеживала изменения в сети, ее строки периодически (через 1...20 минут) принудительно очищаются.

Примечание. *Очевидно, что использование протокола ARP возможно только для сетей, обеспечивающих широковещательную рассылку данных.*

Задачу построения строк таблицы трансляции по известному Ethernet-адресу решает протокол RARP (Reverse ARP), описанный в RFC 903 и использующий сегмент той же структуры, что протокол ARP. Определение IP-адреса по известному Ethernet-адресу требуется в момент начальной загрузки бездисковых ЭВМ, подключенных к сети.

Примечание. *Использование протоколов ARP и RARP может быть отключено системным администратором.*

7.2. Протокол SLIP

Протокол **SLIP (Serial Line Internet Protocol)** обеспечивает соединение двух ЭВМ через последовательный интерфейс (например, **V.24**). Протокол SLIP описан в RFC 1055.

Протокол очень прост. Все SLIP-кадры начинаются со служебного символа 0xEB, называемого ESC, а заканчиваются служебным символом 0xC0, называемым END. Между этими символами располагаются передаваемые данные.

Если служебные символы встречаются в передаваемых данных, то они отсылаются приемнику в виде двухбайтовых последовательностей: {ESC, 0xEC} и {ESC, 0xED}. На принимающей стороне двухбайтовые последовательности преобразуются в ESC и END.

RFC 1055 не специфицирует максимальной длины кадра (MTU), но существующие реализации протокола ориентированы на значение MTU равное 1006 байт.

Примечание. *Очевидно, что скорость передачи данных по последовательному интерфейсу невелика. Для повышения эффективности протокола SLIP в RFC 1144 была предложена его модификация, учитывающая то обстоятельство, что при TCP-взаимодействии по последовательной линии большинство полей IP- и TCP-заголовков остаются неизменными на все время логического соединения. Данная модификация SLIP реально пересылает в своих кадрах только те поля IP- и TCP-заголовков, которые меняют свое значение от кадра к кадру.*

7.3. Протокол PPP

Протокол **PPP (Point-to-Point Protocol)** также может быть использован для соединения двух ЭВМ по последовательному интерфейсу. Протокол PPP (RFC 1331) разработан позднее протокола SLIP, поэтому в нем ликвидированы некоторые недостатки протокола SLIP, в частности:

- поддерживаются различные протоколы вышележащего уровня (а не только IP);
- используются контрольные суммы.

Для идентификации границ PPP-кадра используется служебный символ 0x7E.

Передаче данных по протоколу PPP предшествует этап тестирования и конфигурирования соединения с помощью протокола LCP (Link Control Protocol), являющегося частью PPP. LCP используется и для завершения соединения.

Кроме того, для обмена управляющей информацией используется протокол NCP (Network Control Protocol). Каждый протокол, лежащий выше PPP, имеет свою версию протокола NCP. NCP, определенный для протокола IP, носит название IPCP (Internet Protocol Control Protocol).