# Понятие о стеке протоколов

Допустим перед вами стоит задача передать информацию от пункта А в пункт В. Её можно передавать непрерывно. Но задача усложняется, если надо передавать информацию между пунктами A<-->B и A<-->C по одному и тому же физическому каналу. Если информация будет передаваться непрерывно, то когда С захочет передать информацию в А — ему придётся дождаться, пока В закончит передачу и освободит канал связи. Такой механизм передачи информации очень неудобен и непрактичен. И для решения этой проблемы было решено разделять информацию на порции.  
  
На получателе эти порции требуется составить в единое целое, получить ту информацию, которая вышла от отправителя. Но на получателе А теперь мы видим порции информации как от В так и от С вперемешку. Значит, к каждой порции надо вписать идентификационный номер, что бы получатель А мог отличить порции информации с В от порций информации с С и собрать эти порции в изначальное сообщение. Очевидно, получатель должен знать, куда и в каком виде отправитель приписал идентификационные данные к исходной порции информации. И для этого они должны разработать определённые правила формирования и написания идентификационной информации. Далее слово «правило» будет заменяться словом «протокол».  
  
Для соответствия запросам современных потребителей, необходимо указывать сразу несколько видов идентификационной информации. А так же требуется защита передаваемых порций информации как от случайных помех (при передаче по линиям связи), так и от умышленных вредительств (взлома). Для этого порция передаваемой информации дополняется значительным количеством специальной, служебной информацией.  
  
В протоколе Ethernet находятся номер сетевого адаптера отправителя (MAC-адрес), номер сетевого адаптера получателя, тип передаваемых данных и непосредственно передаваемые данные. Порция информации, составленная в соответствии с протоколом Ethernet, называется кадром. Считается, что сетевых адаптеров с одинаковым номером не существует. Сетевое оборудование извлекает передаваемые данные из кадра (аппаратно или программно), и производит дальнейшую обработку.  
  
Как правило, извлечённые данные в свою очередь сформированы в соответствии с протоколом IP и имеют другой вид идентификационной информации — ip адрес получателя (число размером в 4 байта), ip адрес отправителя и данные. А так же много другой необходимой служебной информации. Данные, сформированные в соответствии с IP протоколом, называются пакетами.  
  
Далее извлекаются данные из пакета. Но и эти данные, как правило, ещё не являются изначально отправляемыми данными. Этот кусок информации тоже составлен в соответствии определённому протоколу. Наиболее широко используется TCP протокол. В нём содержится такая идентификационная информация, как порт отправителя (число размером в два байта) и порт источника, а так же данные и служебная информация. Извлечённые данные из TCP, как правило, и есть те данные, которые программа, работающая на компьютере В, отправляла «программе-приёмнику» на компьютере A.  
  
Вложность протоколов (в данном случае TCP поверх IP поверх Ethernet) называется стеком протоколов.

# Сокеты, IP-адрес и порт

Взаимодействие по сети происходит не абстрактно между компьютерами, а между конкретными программами, которые должны точно знать, куда и как им обращаться. Такое взаимодействие возможно с помощью сокетов.

Сокет - это один конец двустороннего канала связи между двумя программами, работающими в сети.

Сокеты бывают 2-х типов:

1. Потоковые.

Сокеты с установленным соединением (базируются на протоколе TCP). Состоят из потока байтов, который может быть двунаправленным - приложение может и передавать, и получать данные.

1. Дейтаграммные.

Сокеты, не требующие установления явного соединения между ними (базируются на протоколе UDP). Сообщение отправляется указанному сокету и, соответственно, может получаться от указанного сокета.

Сокет состоит из IP-адреса и порта.

1. IP-адрес - уникальный сетевой адрес узла в компьютерной сети, построенной по протоколу IP.

В версии протокола IPv4 IP-адрес имеет длину 4 байта (например, 192.168.0.3), а в версии протокола IPv6 IP-адрес имеет длину 16 байт (например, 2001:0db8:85a3:0000:0000:8a2e:0370:7334).

В любой сети требуется уникальность адреса.

Существует ряд специальных IP-адресов, например, 127.0.0.1 - IP-адрес, с помощью которого компьютер может обратиться по сети к самому себе, независимо от наличия у него подключения к сети, вида сети и адреса компьютера в ней.

1. Порт - натуральное число, записываемое в заголовках протоколов транспортного уровня (TCP, UDP и др.).

Порт используется для определения процесса-получателя пакета в пределах одного хоста. Компьютер, на котором в одно время выполняется несколько приложений, получая пакет из сети, может идентифицировать целевой процесс, пользуясь уникальным номером порта, определенным при установлении соединения.

За определенными службами номера портов зарезервированы - это широко известные номера портов, например, порт 80, использующийся в протоколе HTTP. Любое приложение может пользоваться любым номером порта, который не был зарезервирован и пока не занят. Агентство IANA (англ. Internet Assigned Numbers Authority - Администрация адресного пространства Интернет) ведет перечень широко известных номеров портов.

Поскольку IP-адрес уникален в пределах сети, а номера портов уникальны на отдельной машине, номера сокетов также уникальны во всей сети (в т.ч. сети Интернет). Эта характеристика позволяет процессу общаться через сеть с другим процессом исключительно на основании номера сокета.

Принципы работы стека протоколов TCP/IP относительно просты и напоминают работу обычной почты, где:

* пакет: бумажное письмо - содержит передаваемые данные и адресную информацию (адрес отправителя и адрес получателя);
* IP-адрес: адрес дома;
* номер порта: номер квартиры.

На почте для отправки письма каждый придерживается следующей схемы:

* пишет на листе письмо, кладет его в конверт, заклеивает;
* на обратной стороне конверта пишет адреса отправителя и получателя;
* относит конверт в ближайшее почтовое отделение;
* в момент доставки письмо проходит через цепочку почтовых отделений до ближайшего почтового отделения получателя, откуда почтальоном доставляется по указанному адресу получателя и опускается в его почтовый ящик (с номером его квартиры) или вручается лично - письмо дошло до получателя.

Когда получатель письма захочет ответить, то он в своем ответном письме поменяет местами адреса получателя и отправителя, и письмо отправляется по той же цепочке, но в обратном направлении.

# Клиент-серверная Архитектура

«Клиент — сервер» (англ. client–server) — архитектура, в которой сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине).

Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

# Порядок следования байт (endianness)

Понятие Byte order, или порядок следования байт (или **endianness**) относится к очередности размещения в памяти многобайтовых величин (обычно целых чисел и чисел с плавающей точкой; хотя числа с плавающей точкой не используются в Linux kernel, они могут работать в пользовательских программах), как это поддерживается аппаратурой процессора. Соответственно бывает 2 варианта порядка байт - **big endian** и **little endian**. Big endian это такой порядок байт, когда самый значимый по значению байт числа (most significant byte) сохранен в памяти первым по порядку (т. е. у него самый маленький абсолютный адрес байта, в сравнении с остальными байтами числа). Соответственно little endian это противоположный порядок байт, когда наименее значимый байт сохраняется в памяти первым.

Чтобы было понятнее, рассмотрим пример. 4-байтное целое число 0x01020304 будет сохранено в памяти системы big endian следующим образом:

|  |  |  |  |
| --- | --- | --- | --- |
| **Байт0** | **Байт1** | **Байт2** | >**Байт3** |
| 0x01 | 0x02 | 0x03 | 0x04 |

Big endian всегда используется для так называемого **сетевого порядка байт** (network byte order), который применяется при кодировании адресов в сетевых протоколах.

Та же самая величина, которая будет храниться в памяти системы little endian, разместится в противоположном порядке:

|  |  |  |  |
| --- | --- | --- | --- |
| **Байт0** | **Байт1** | **Байт2** | **Байт3** |
| 0x04 | 0x03 | 0x02 | 0x01 |

Обычно при программировании можно не обращать внимания на endianness, то есть не важно, как процессор сохранит байты чисел в системе - big endian или little endian; ядро CPU просто загружает данные из памяти и сохраняет данные в память, и представляет данные в Вашей программе уже в правильном виде. Однако, когда нужно обмениваться данными с другой системой, обе системы должны учитывать формат хранения данных в памяти (endianness).

Linux kernel может быть либо big endian, либо little endian, в зависимости от архитектуры, в расчете на которую kernel скомпилировано. Ниже в таблице показан endianness для различных типов архитектур процессоров и протоколов.

|  |  |  |
| --- | --- | --- |
| **Big Endian** | **Little Endian** | **Оба варианта** |
| **Архитектуры процессоров** | | |
| AVR32 FR-V H8300 PA-RISC S390 Motorola 680x0 PowerPC SPARC | Alpha CRIS Blackfin Intel 64 IA-32 (x86) MN10300 AT91SAM7 Cortex, STM32 | ARM SuperH (sh) M32R MIPS Xtensa |
| **Протоколы** | | |
| TCP/IP | USB |  |

Примечание: процессор ARM может быть либо с архитектурой big endian, либо little endian, в зависимости от типа применяемого чипа, однако чаще всего это big endian. Архитектура PowerPC может быть сконфигурирована для работы либо в режиме big endian, либо little endian, но в Linux используется только big endian.

## ****Почему следует беспокоиться об endianness****

Как уже упоминалось, порядок байт на низком уровне в системе полностью прозрачен для программиста (про endianness можно часто забыть). Однако могут быть проблемы, когда происходит обмен данными с другой системой, поскольку эта другая система может ожидать появления данных в блоке памяти в противоположном порядке.

Например, если заранее нельзя предсказать тип системы на каком-то дальнем окончании сетевого соединения, сетевые протоколы должны заранее определить порядок байт, используемый для хранения многобайтных величин в заголовках пакетов. В этом случае порядок байт называют сетевым порядком байт (network byte order), и для протокола **TCP/IP** это будет big endian. Таким образом, отправляющая пакеты система конвертирует данные из локального порядка хранения байт в сетевой. После этого принимающая система преобразует данные из сетевого порядка байт в локальный. На практике, когда есть жесткие требования к быстродействию и заранее известно, что локальный порядок байт такой же, как сетевой, операция конверсии отбрасывается в целях оптимизации.