Глава 1
СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ, СОКЕТЫ, ПОРЯДОК БАЙТОВ

> Никогда не недооценивайте пропускную способность несущегося по шоссе грузовика, набитого магнитными картриджами.
>
> *Эндрю Таненбаум, «Компьютерные сети» (изд-во «Питер»)*

Введение

Сперва ответим на вопрос, что же такое сетевое программирование.

Сетевое программирование — это реализация обмена данными по сети между определенными сущностями. Сеть в данном случае — среда для связи.

> **Компьютерная сеть** — это множество взаимодействующих и совместно использующих ресурсы вычислительных устройств. Сеть состоит из узлов и каналов, соединяющих узлы.

Такую среду можно представить в виде графа, изображенного на рис. 1.1, где узлы — это субъекты взаимодействия, а ребра — каналы обмена данными между узлами.

Субъектами взаимодействия могут быть:

- физические устройства;
- драйверы протоколов;
- приложения;
- пользователи, которые общаются по сети, используя данные приложения.

Каналы могут быть физическими, например проводной Ethernet или радиоканал, либо виртуальными, например канал OpenVPN, опирающийся на протоколы более высокого уровня, или разделяемые на маршрутизаторе по тегам — каналы VLAN.

В этой главе мы:

- разберем, что такое сетевое программирование, зачем оно нужно и какое место занимает в структуре знаний и навыков специалиста;

- узнаем, какие стандарты определяют интерфейс сокетов и работу протоколов;
- узнаем, где и как находить информацию;
- поймем, что такое сокеты, как их создавать, закрывать и назначать им адреса.

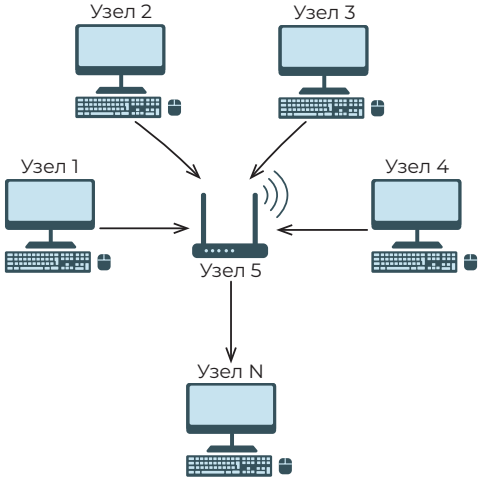

Рис. 1.1. Схема обмена данными по сети

Чтобы закрепить знания, мы начнем разрабатывать свою простую обертку над сокетами.

> Когда мы говорим о сетевом программировании, то в большинстве случаев подразумеваем написание сетевых приложений, работающих под управлением ОС. Но это не всегда так. Реализация сетевых протоколов, например их стеков, также относится к области сетевого программирования.
>
> В книге мы рассматриваем преимущественно разработку приложений.

Модели сетевого взаимодействия

Сетевое взаимодействие приложений представлено стеком уровней. Разбивка по уровням дает возможность использовать на каждом из них независимые протоколы, не затрагивая вышележащие.

Физический уровень отправляет единицы и нули по кабелю, оптоволокну, радиоканалу и т. п.

Затем уровень канала передачи данных организует эти единицы и нули во фрагменты данных и безопасно доставляет их в нужное место по сети. Сетевой уровень передает организованные данные по нескольким сетям, а транспортный уровень доставляет данные нужному приложению в пункте назначения.

Стек уровней в компьютерных сетях описывается такими моделями, как OSI — Open Systems Interconnection или DoD — Department of Defence — модель Министерства обороны США.

Модель OSI, показанная на рис. 1.2, — наиболее общая. Она состоит из семи уровней, каждый из которых обычно представлен одним протоколом либо его частью.

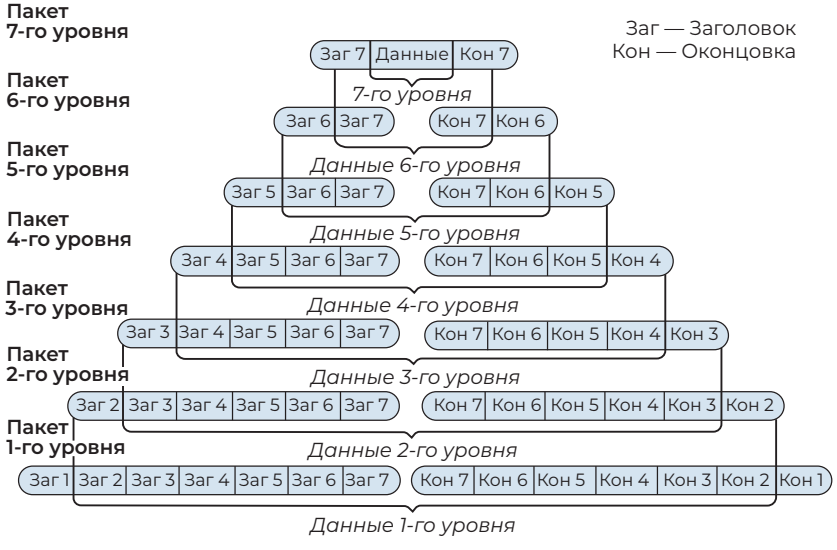

Рис. 1.2. Модель OSI

Уровни OSI сверху вниз:

- **Прикладной** — протоколы, которые используют конкретные приложения, например браузер: HTTP, FTP и т. п.
- **Представления данных** — кодировки ASCII, UNICODE, преобразования UTF, форматы данных, например JPEG или PNG, и т. п.
- **Сеансовый** — протоколы установления сессии, аутентификации, такие как L2TP, PAP/CHAP и т. п.
- **Транспортный** — обеспечивает передачу данных внутри сети: TCP, UDP, UDP Lite, SCTP и т. д.
- **Сетевой** — обеспечивает возможность обмена данными между разными сетями: IPv4, IPv6, устаревший — IPX.
- **Канальный** — протоколы обмена данными по каналу. Например Ethernet, определяющий формат кадров и правила согласования настроек адаптеров, PPP, выполняющий согласование параметров обмена, в частности PPPoE, работающий поверх Ethernet.
- **Физический** — различные физические каналы, спецификация типа среды, формы сигналов, напряжений, если это применимо к среде, например Ethernet по витой паре.

Модель OSI критикуют за то, что не все ее уровни всегда явно присутствуют в реальных системах.

Уровней в ней действительно много, и последние три часто описываются и реализуются в рамках одного протокола.

На практике удобнее работать с более простыми моделями, которых около десятка: модель Таненбаума, модель Cisco Academy и др. Одна из таких упрощенных моделей, показанная на рис. 1.3, — это DoD-модель, она же модель TCP/IP, которая содержит всего четыре уровня:

- **Прикладной уровень** — объединение прикладного и сеансового уровней и уровня представления OSI. Уровень конкретных протоколов, используемых приложениями.

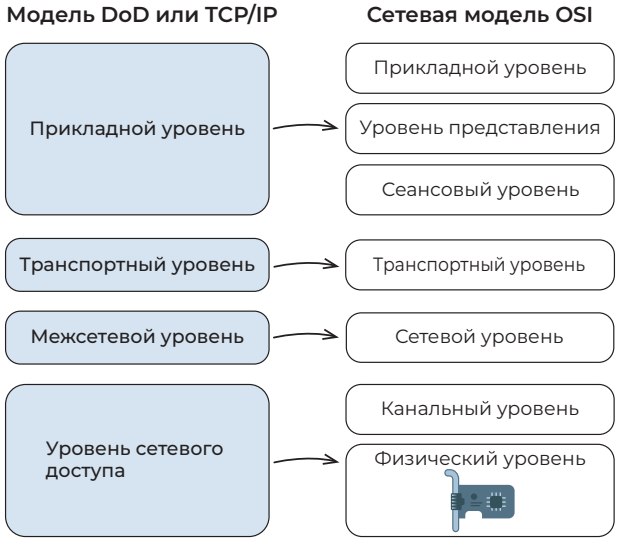

Рис. 1.3. DoD-модель

- **Транспортный уровень** — соответствует транспортному уровню OSI. Отвечает за гарантированную или негарантированную доставку и правильную последовательность прихода данных.
- **Межсетевой** — аналог сетевого уровня OSI. Нужен для передачи данных из одной сети в другую, независимо от протоколов нижнего уровня.
- **Уровень сетевого доступа** — объединяет физический и канальный уровни OSI. Описывает способ кодирования данных на физическом уровне и среду передачи, ее физические параметры.

Количество уровней вложенности потенциально не ограничено. Как правило, новые протоколы добавляются поверх модели.

> В некоторых случаях, например при использовании отдельных реализаций VPN, поверх транспортного или прикладного уровня может «вырасти» стек протоколов, начиная с уровня верхнего протокола.

Обмен данными по сети

В моделях данные переходят с верхних уровней на нижние и инкапсулируются в обертки протоколов, необходимые для сетевого обмена.

Эти обертки в рамках модели называются **служебными модулями данных** — Service Data Unit, или **SDU**. Служебные потому, что передаются между уровнями внутри модели.

С последнего уровня производится отправка данных в сеть.

После того как нижний уровень принял данные, он «снимает» обертку, которая предназначена для него, и передает данные, которые были обернуты, верхнему уровню. По сути, данные на предыдущем уровне декапсулируются обратно в понятные следующему уровню.

Получается, что реальное взаимодействие происходит на самом нижнем, физическом уровне. Но если выполнить срез на любом из уровней, понятные ему данные будут передаваться между ним и тем же уровнем на другой стороне коммуникационной среды, как показано на рис. 1.4.

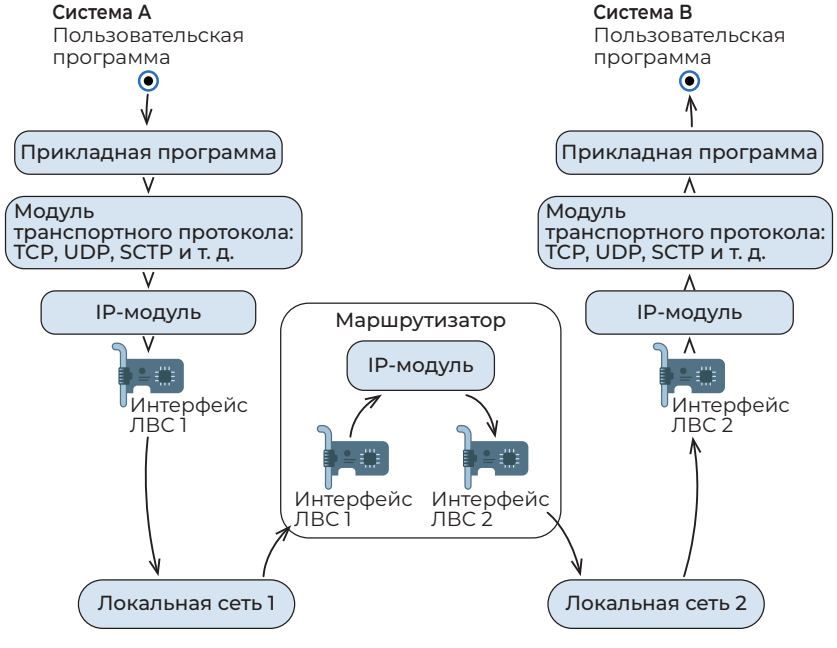

Рис. 1.4. Срез обмена данными по сети

Данные, которые циркулируют между приложениями, на каждом уровне передаются как **единицы данных протокола**, или Protocol Data Unit — **PDU**. Это могут быть пакеты TCP, дейтаграммы или IP-пакеты.

Существуют разные соглашения об именовании IP PDU:
- В MIL-STD-1777 и RFC-791 «Internet Protocol» — **дейтаграмма**.
- Популярное название в книгах — **пакет**.

В этой книге мы будем использовать популярное название «**пакет**».

Для PDU TCP будем использовать название «**сегмент**», а для UDP — «**дейтаграмма**».

Приложение обычно не занимается их обработкой, которая возложена на стек ОС, но может и обрабатывать их, если, например, требуется осуществлять маршрутизацию не так, как это делает операционная система.

Такая возможность позволяет строить нетривиальные схемы работы в сети. Например, VPN, работающий поверх HTTPS, позволяет обходить DLP-системы провайдеров, которые блокируют иные протоколы. Другой пример — разрабатываемый стандарт DoH, который подразумевает передачу DNS-запроса по обычному защищенному HTTPS-протоколу.

Протоколы нижних уровней в общем случае могут передавать любые данные, но внешние требования, которые реализуются в приложениях, часто ограничивают их, задавая следующие параметры:

- время задержки от начала передачи;
- скорость передачи;
- возможность потерь и гарантии доставки;
- ограничение на маршрут передачи, особенно если данные конфиденциальные;
- срочность доставки.

Некоторые требования противоречивы; например, гарантия доставки с пересылкой увеличивает надежность, однако уменьшает скорость передачи данных.

Поэтому для разных задач есть транспортные протоколы, как правило, реализуемые в стеке операционной системы:

- **TCP** — протокол, обеспечивающий гарантированную упорядоченную надежную передачу данных. Позволяет организовать поток из одного узла в другой. При его использовании отправитель точно знает, получил адресат данные или нет. Надежность в данном случае означает, что данные точно придут без искажений в содержании. Поток байтов означает, что отдельные вызовы отправки данных не создают границ и могут быть преобразованы во множество PDU. Этот протокол был описан и реализован в 1974 году.
- **UDP** — протокол без гарантий. На его основе пользователи могут реализовывать собственные протоколы. В рамках протокола UDP производится обмен сообщениями или дейтаграммами, каждый вызов отправки данных создает один PDU — дейтаграмму, которая содержит точно одну порцию данных.
- **UDP Lite** — протокол UDP с частичным расчетом контрольной суммы. Является расширением протокола UDP и используется для того, чтобы не выполнять лишние вычисления.

- **SCTP** — Stream Control Transmission Protocol. Транспортный протокол, как и TCP, обеспечивающий надежную упорядоченную передачу данных. Протокол был разработан в 2000 году, в нем исправлены недостатки TCP и реализованы многопоточность (внутри одного соединения может быть несколько каналов), защита от DDoS атак и синхронное соединение между двумя хостами по двум и более независимым физическим каналам. Обмен данными производится в виде сообщений, границы которых сохраняются.

Это не единственные транспортные протоколы, мы не охватили еще достаточно многие, но основные — это TCP и UDP, поверх которых реализуется большинство других.

> Протокол SCTP, который был призван устранить недостатки TCP, так и не стал популярным. Он плохо поддерживается сетевым оборудованием и редко используется на практике.

Кроме транспортных протоколов общего назначения, существуют специальные транспортные протоколы для особых условий, и позже мы вкратце поговорим и о них.

> Протокол удаленного сетевого обмена TCP был изобретен Робертом Эллиотом Каном совместно с Винтоном Серфом в рамках решения проблемы совместимости систем и каналов связи. Р. Э. Кана и В. Серфа называют отцами-основателями интернета.

В DoD-модели уровнем выше транспортного реализуются протоколы для обмена данными между приложениями.

Адресация в протоколах

Некоторые протоколы вводят свою адресацию, например:

- **Протоколы сетевого уровня**, например IP, вводят адреса узлов и сетей.
- **Протоколы канального уровня**, например Ethernet, вводят адреса оборудования, или MAC-адреса, для того чтобы доставить кадры на правильный узел из множества подключенных к шине. MAC-адрес используется даже в соединениях типа «точка-точка», хотя в этом случае он является просто рудиментом.
- **Протоколы транспортного уровня**, такие как TCP и UDP, вводят идентификаторы портов для того, чтобы выбрать, какому процессу на узле отправить принятые данные.

Сказанное выше означает, что на разных уровнях модели появляются разные адреса и сеть больше нельзя полностью описать только одним графом.

Для каждого уровня с адресацией имеется свой граф, описывающий топологию. Но в частном случае, когда протокол некоторого уровня не предполагает адресации, графы данного уровня и следующего за ним могут совпадать.

Например в сети Ethernet граф физического уровня будет состоять из узлов, подключенных к одной шине, один из которых — маршрутизатор, подключенный к другому маршрутизатору через соединение «точка-точка», а к тому, в свою очередь, подключены узлы на одной шине.

На сетевом уровне эта топология может быть представлена двумя подсетями — каждая за своим маршрутизатором. Шина видна не будет, то есть граф будет состоять из двух соединенных «звезд» с центрами-маршрутизаторами.

Если же сети объединены маршрутизаторами на канальном уровне в одну сеть, их граф сетевого уровня будет представлять собой просто связь всех со всеми, а маршрутизаторы вообще не будут видны.

Примеры топологий представлены на рис. 1.5.

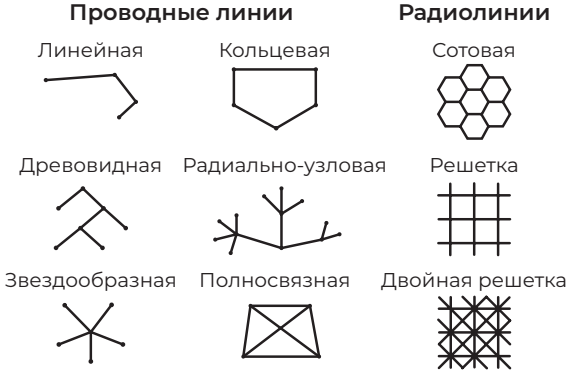

Рис. 1.5. Топологии сетей

Обычно физическая сеть представлена шиной, то есть полносвязным графом, где все связаны со всеми; однонаправленным кольцом, например, с передачей маркера; или топологией «звезда», когда все узлы подключены к разным портам центрального роутера, а он решает, кому, что и куда отправлять.

Однако на уровне коммуникации приложений у каждого узла может быть доступ ко всем остальным узлам, то есть сеть будет полносвязной.

Задачи, которые решает книга

В книге мы будем писать код для компьютерных сетей, которые имеют следующие особенности:

- **Это цифровые сети.** В них, независимо от среды передачи, с физического уровня поступают нули и единицы и любой сигнал оцифровывается.
- **Это сети, для которых предполагается не канальная, а пакетная коммутация.** Канальная коммутация тоже встречается, например, в случае использования модемов для соединения через PSTN, но даже тогда, при установленных каналах, верхние уровни обеспечивают пакетную коммутацию.

Пакетная коммутация настолько хорошо себя зарекомендовала, что используется даже в сетях межпланетной связи.

- **Абсолютное большинство таких сетей стандартизированы и основаны на стеке TCP/IP.** Когда-то для локальных сетей был популярен стек IPX/SPX, но с угасанием популярности ОС Novell NetWare он уступил место TCP/IP и больше практически не встречается.

Большую часть внимания в книге мы уделим следующим вопросам:

- изучению и использованию того, что реализовано выше транспортного уровня OSI;
- изучению API и библиотек;
- реализации собственных простых протоколов прикладного уровня.

API, предоставляемый операционной системой, обычно позволяет работать поверх уровня транспортного протокола и частично конфигурировать нижележащие уровни для обмена данными.

Обычно непривилегированному пользователю запрещено работать на уровнях ниже транспортного, потому что это небезопасно. Но бывают случаи, когда работа непривилегированного пользователя на уровнях ниже транспортного оправданна:

- **Прослушивание всего трафика**, который приходит в сетевые устройства, и обычно его последующая интерпретация либо запись. Используется в анализаторах трафика, или снифферах.
- **Приложения реального времени**, которые должны знать, сколько времени требуется на запрос и ответ. В ОС, которые не являются системами реального времени, зачастую недопустимы задержки, которые вносит стек ОС. Как пример — VoIP-приложения, которые могут работать поверх сетевого уровня или даже ниже, самостоятельно формируя дейтаграммы и пакеты.
- **Различные специальные протоколы**, например, требующие отправки большого потока трафика, такие как протоколы синхронизации между узлами кластеров.
- **Экспериментальные реализации сетевых протоколов**, в том числе протоколов канального уровня: PPP, MPLS и др.
- **Конфигурирование и настройка** сетевого оборудования.
- **Использование служебных протоколов**: ICMP, IGMP, SNMP и др.
- **Реализация взаимодействия** с операционными системами, которые не поддерживают TCP/IP. Например, QNX 4.25 использует собственные протоколы. А простейшим сетям на микроконтроллерах зачастую не хватает ресурсов памяти и CPU, чтобы выполнять полный код стека со всеми проверками и условиями.

Все эти случаи мы рассмотрим в книге и некоторые из них — подробно.

Служебные протоколы, используемые в стеке TCP/IP, мы разберем отдельно. Прежде всего поговорим о протоколе ICMP — Internet Control Message Protocol, который позволяет узнавать состояние отдельных узлов сети.

В остальном даже протоколы, необходимые для взаимодействия маршрутизаторов, реализуются поверх TCP, как, например, BGP, или поверх IP, как например, OSPF. И поэтому работе с протоколами, реализованными поверх TCP и IP, мы уделим больше внимания.

Ресурсы, необходимые для работы

Ни одна книга не может быть всегда актуальной и такой же точной, как спецификации и код.

Обычно книга дает картину «с высоты», а при увеличении масштаба неизбежны «белые пятна». Но книга может подсказать, как и где найти подробную информацию.

Перечислим некоторые источники данных о сетевых протоколах и способах их реализации. На другие источники мы будем ссылаться по ходу изложения.

RFC — Request For Comments

Это предложения изменений, которые де-факто являются стандартом после их публикации, хотя де-юре им не являются.

Вот некоторые примеры:

- RFC 1 «Host Software»[1] — программное обеспечение узла. Первое, уже не вполне актуальное RFC.
- RFC 791 «Internet Protocol»[2] — описание IP.
- RFC 768 «User Datagram Protocol»[3] — описание UDP.
- RFC 793 «Transmission Control Protocol»[4] — описание TCP.
- RFC 1011 «Official Internet Protocols»[5] — описание стандартных протоколов Internet.
- RFC 1180 «A TCP/IP Tutorial»[6] — туториал по TCP/IP.
- RFC 2553 «Basic Socket Interface Extensions for IPv6»[7] — расширения сокетного API для IPv6.
- RFC 8085 «UDP Usage Guidelines»[8] — правила использования UDP.

В RFC содержится практически вся информация, необходимая для понимания и реализации протоколов, а значит, и сетевого взаимодействия. Но зачастую их тяжело читать. К тому же они дают лишь понимание того, как работает протокол, но не о том, какие реализации существуют, если только это не RFC о реализациях. Таким образом, RFC — это отправная точка для тех, кто реализует протокол, и тех, кто хочет точно понимать все нюансы.

Прикладные разработчики читают RFC нечасто, хотя в них описано большинство действующих стандартов интернета.