Глава 3. Протоколы и модели

В данной главе будут рассмотрены:

- Правила обмена сообщениями;
- Способы рассылки сообщений;
- Сетевые модели и протоколы.

3.1. Обмен данными

Обмен данными является изначальной и главной целью создания и существования компьютерных сетей. Ежеминутно миллиарды устройств отправляют и получают информацию. Подобный обмен был бы невозможен без «понимания» этими устройствами правил взаимодействия в локальных и глобальных сетях.

Цель данной главы – познакомить с работой основных правил взаимодействия в сети – *протоколов*. Существует большое количество протоколов: одни из них отвечают за весьма узкую специализацию, другие применяются глобально, одни применяются только в оборудовании одного определенного производителя *(проприетарные протоколы)*, другие приняты и утверждены инженерным сообществом в качестве стандартов (*стандартизированные протоколы*).

3.1.1. Установление правил обмена данными

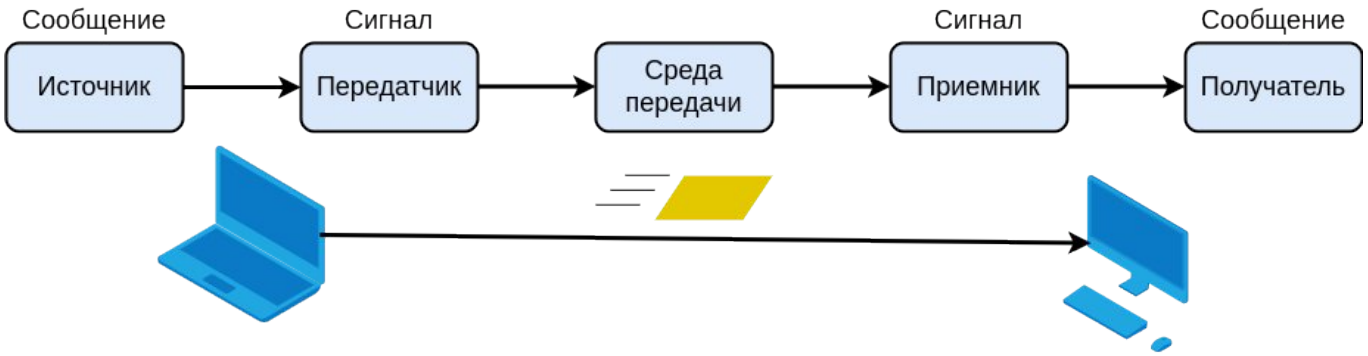


Рисунок 3.1.1.1. – Структурная схема процесса передачи данных от передатчика к приемнику

Для того чтобы один пользователь мог передать сообщение другому пользователю, необходимо соблюдение ряда условий:

1) Отправитель знает, кому он отправляет сообщение, то есть может идентифицировать получателя;
2) Отправитель имеет техническую возможность отправить сообщение. Имеется в виду, что устройство отправителя может принять само сообщение из приложения пользователя, преобразовать его в сигналы определенного формата и отправить их через сеть в направлении получателя. Если сообщение не может быть передано одним «куском», необходимо его разбить на фрагменты (осуществить *фрагментацию*).
3) Сеть, к которой подключено устройство отправителя, принимает сигналы этого устройства и может их передать либо сразу на устройство получателя, либо в другую сеть по направлению к получателю;

4) Сеть, к которой подключено устройство получателя, может преобразовать сообщение в сигналы, которые «понимает» устройство получателя. То есть, сеть должна быть правильно настроена и работать на одних и тех же протоколах, чтобы приемная и передающая стороны «понимали» друг друга.

5) Устройство получателя может принять сигналы из собственной сети, расшифровать их (при необходимости собрать сообщение из *фрагментов*) и *презентовать* сообщение получателю в выбранном получателем *приложении*.

Протоколы устанавливают правила обмена информацией, включая кодирование сообщений, варианты доставки и синхронизации, форматирование, инкапсуляцию и размер сообщений. На различных этапах обмена информацией могут использоваться специализированные протоколы, которые передают результат работы другим протоколам.

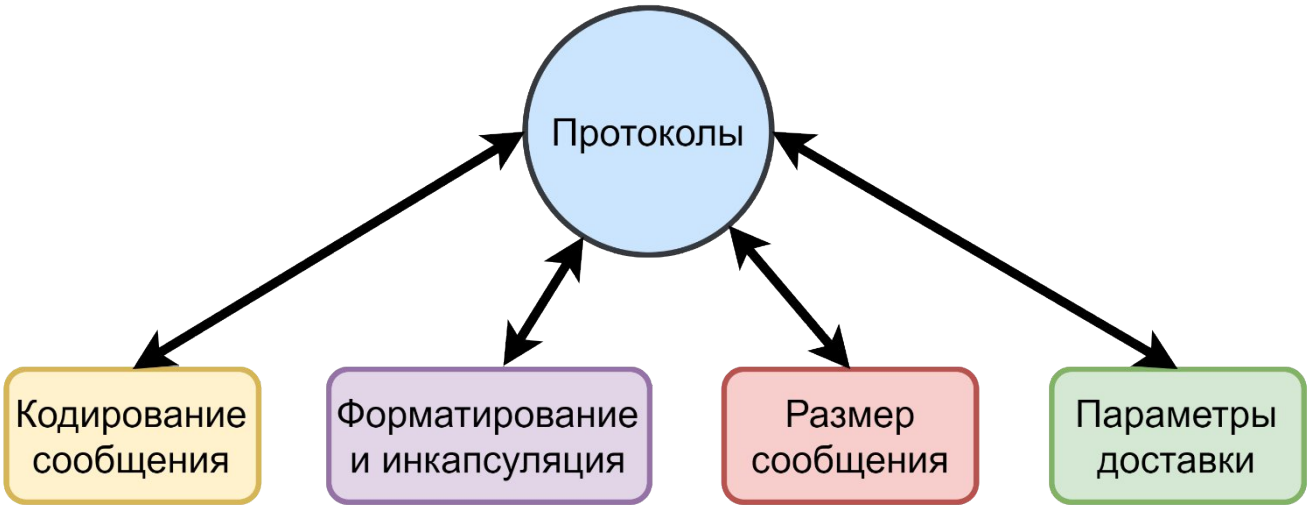


Рисунок 3.1.1.2. – Задачи протоколов

3.1.2. Кодирование сообщения

В общем смысле кодирование – преобразование информации из одного формата в другой, более подходящий для той или иной цели. Для «общения» в сети оконечных и сетевых устройств наиболее удобен формат двоичных сигналов, поэтому все сообщения кодируются именно в них. Однако для разных сред передачи данных приходится перекодировать сообщения заново, так как сигналы для кабельной среды (электрические или световые импульсы) отличаются от сигналов для беспроводной среды.

В информационных сетях кодирование может происходить на разных этапах передачи информации, начиная с преобразования текста в двоичный код и заканчивая превращением кода в электрические импульсы. Правила кодирования устанавливаются протоколами, которые будут рассмотрены далее в этой главе.

Также, в вопросе кодирования сообщений необходимо сказать про модуляцию сигнала. Модуляция - это изменение одного или нескольких параметров высокочастотного несущего сигнала, в соответствии с низкочастотным информационным сигналом. То есть, низкочастотный сигнал, в котором содержится полезная нагрузка, накладывается на несущий высокочастотный сигнал. Это делается для того чтобы сформировать сигнал с полезной нагрузкой, который способен передаваться по среде передачи данных.

Рассмотрим, как это действует в совокупности. Допустим, мы хотим передать какое-то число по каналу связи. Сначала мы это число кодируем то есть, преобразуем число из десятичной системы счисления, например, в двоичную. Затем в дело вступает модуляция, которая преобразует 0 и 1 в сигналы различной частоты (если используется частотная модуляция), которые мы уже можем передавать через среду передачи.

Разберем на конкретном примере. Мы хотим передать через среду передачи число 149. У нас есть это число, и передатчик, который способен создавать сигнал высокой частоты и заданной амплитуды. Но сам по себе этот сигнал не несет в себе информации. Для этого мы число 149 переводим в двоичную систему счисления, получаем 10010101. Теперь мы говорим, что 1 соответствует сигналу с единичной амплитудой. А 0 соответствует сигналу с нулевой амплитудой. То есть, 1 – есть сигнал, 0 – нет сигнала. Таким нехитрым образом мы передали двоичную последовательность сквозь среду передачи, где на приемной стороне её могут перевести обратно в десятичную систему счисления, и получить исходное число 149.

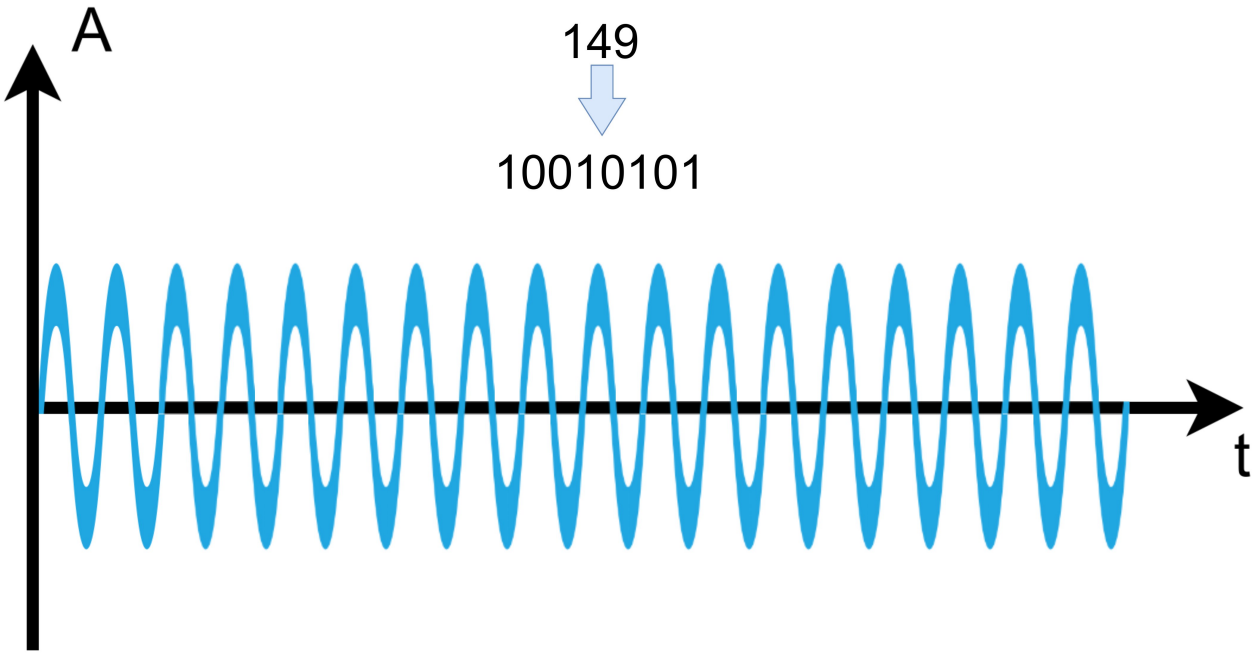


Рисунок 3.1.2.1. – Пример высокочастотного сигнала.

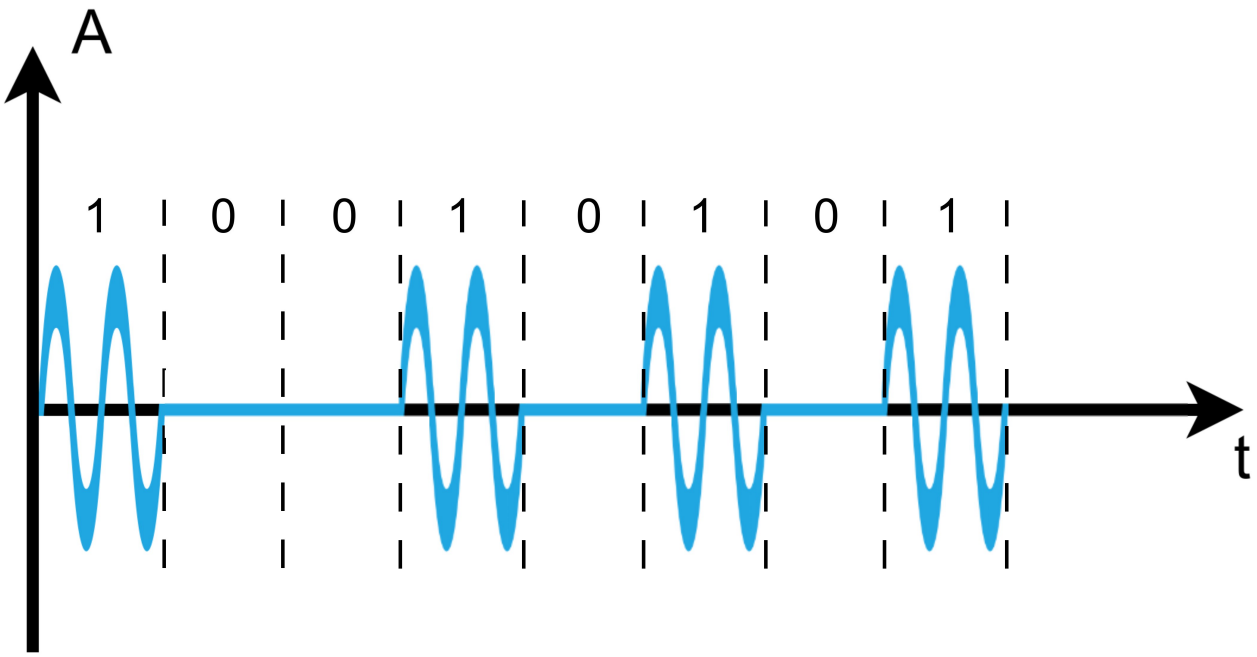


Рисунок 3.1.2.2. – Пример модулированного сигнала.

3.1.3. Форматирование сообщения

Кодирование не может быть односторонним процессом, в противном случае сообщение не будет раскодировано устройством назначения. Передаваемая информация должна быть принята и понята получателем, следовательно, вместе с сообщением необходимо передать дополнительные данные, указывающие на те или иные параметры кодировки и работы протокола.

3.1.4. Размер сообщения

Как люди разбивают длинные сообщения на маленькие предложения, так и компьютеры ограничивают максимальный объем информации, передаваемой за один раз. Это может быть сделано для экономии ресурсов сети, оптимизации работы узлов или обеспечения обратной совместимости со старыми стандартами.

3.1.5. Временные параметры сообщения

К временны*м* параметрам можно отнести скорость передачи сообщения, время начала передачи и время, в течение которого ожидается ответ.

Понимание, когда сеть свободна для отправки информации, позволяет обеспечить доставку и эффективно реагировать на возможные инциденты.

Знание того, с какой скоростью узел может обработать принимаемую информацию, позволяет оптимизировать входящую нагрузку.

При наличии информации о предполагаемом времени ответа, можно повторно отправить сообщение, если ответ не был получен.

3.1.6. Способы рассылки сообщения

Существует три основных способа рассылки сообщений: одноадресная, групповая и широковещательная.

Одноадресная рассылка (англ. *unicast*), как следует из названия, осуществляется от одного отправителя явно определенному получателю. Другие получатели это сообщение получить не могут. Точность передачи определяется и обеспечивается соответствующей *адресацией.* То есть отправитель посылает сообщение на определенный адрес, принадлежащий конкретному получателю.

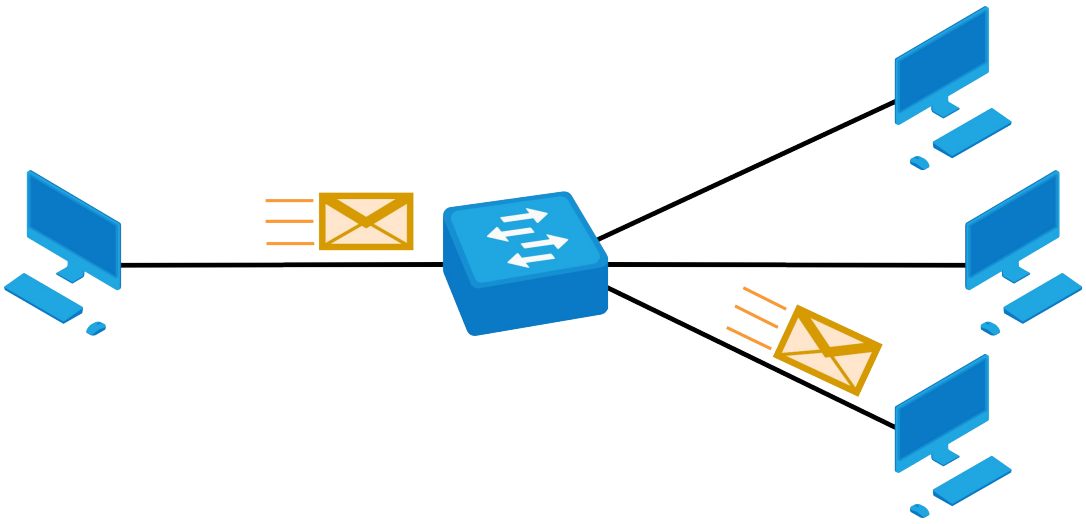


Рисунок 3.1.6.1. – Одноадресная рассылка

Второй метод рассылки — групповая рассылка (англ. multicast). Отправитель посылает сообщение на определенный адрес, который могут «прослушивать» сразу несколько получателей. В связи с этим данный вид доставки иногда называют «один – группе», а соответствующий адрес – групповым адресом.

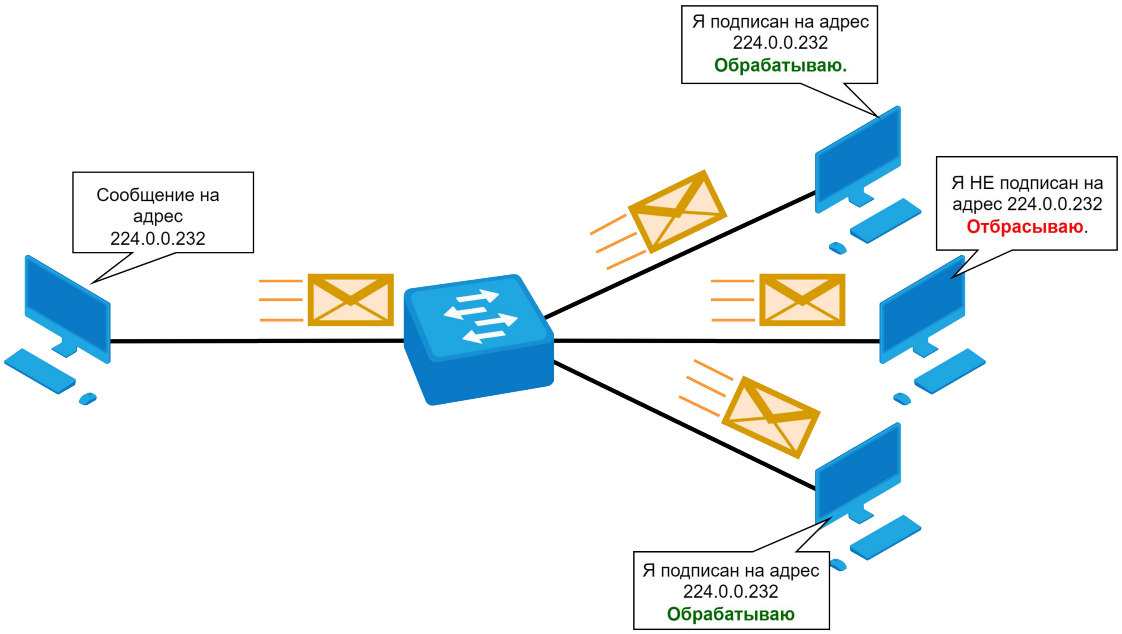


Рисунок 3.1.6.2. – Групповая рассылка

Третий из основных методов рассылки – широковещательная рассылка (англ. *broadcast*). Такие сообщения отправляются на специальные широковещательные адреса и обрабатываются всеми устройствами в сети.

К широковещательным рассылкам в сети необходимо подходить с разумной осторожностью. Дело в том, что большое их количество сильно снижает

работоспособность сети. Но и совсем запретить их невозможно, так как работа некоторых протоколов основана на принципе «спросить у всех, кто-нибудь ответит» (протоколы ARP и DHCP, в которых используется широковещательная рассылка, рассмотрены далее)

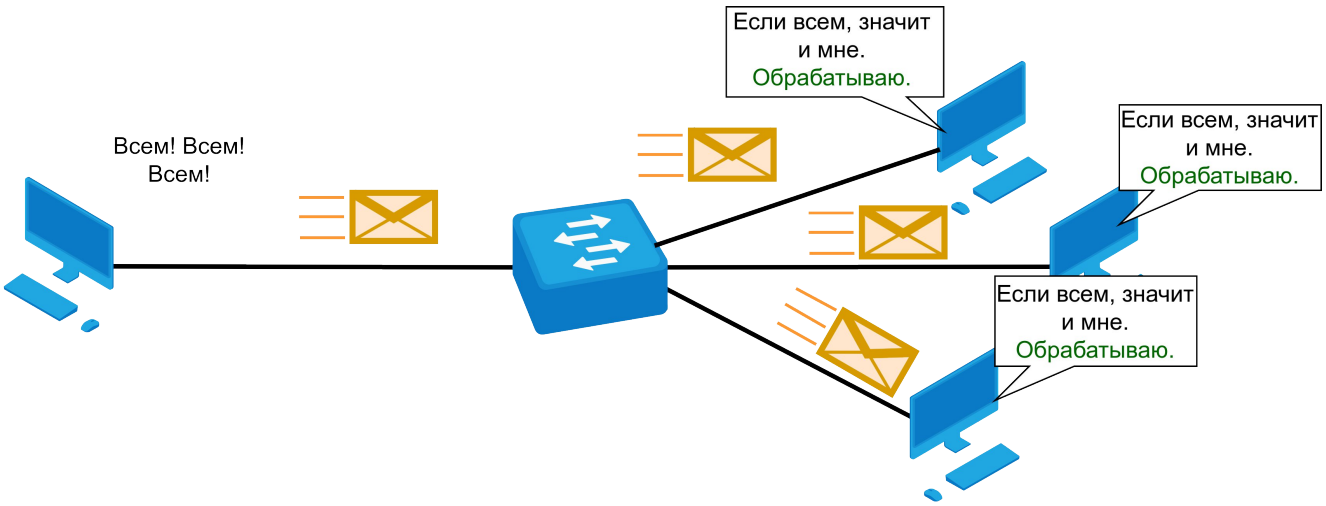


Рисунок 3.1.6.3. – Широковещательная рассылка

Таблица 3.1.6.1. – Типы рассылки сообщений и их получатели

Тип рассылки	Кто получает	Кто обрабатывает
Одноадресная	Один узел	Один получатель
Групповая	Все узлы	Группа подписчиков
Широковещательная	Все узлы	Все узлы

3.1.7. Сегментация и мультиплексирование

Отправитель отправляет сообщение со своего устройства с помощью приложения. А что произойдет, если таких приложений больше одного? Простейший пример: у вас на смартфоне есть и WhatsApp, и Telegram, и в каждом из них вы общаетесь с определенными людьми. Вы же не ожидаете, что сообщение, которое вам посылает ваш контакт *А* в Telegram попадет вашему контакту *Б* в WhatsApp? Правильно, потому что устройство определяет, какое приложение отправило конкретное сообщение, и, более того, способно выделять трафик для нужного приложения из входящего трафика и направлять его соответствующим образом.

Вопрос: как оно это делает?

Дело в том, что при отправке сообщения приложением, другой протокол, работающий на устройстве, делит его на удобные для отправки части (сегментирует), и каждую такую часть (сегмент) помечает определенной меткой, соответствующей приложению или протоколу, создавшему это сообщение *(порт источника)* и приложению или порту, для которого этот сегмент предназначается *(порт назначения)*. Метка сегмента передается вместе с сообщением через все возможные сети до получателя. Когда получатель отвечает на сообщение, то аналогичный протокол на его устройстве помечает сообщение как предназначенное для соответствующего приложения. И когда ответное сообщение приходит к отправителю, оно передается устройством нужному приложению согласно метке сегмента *(номеру порта)*.

Итак, *сегментация* – это разделение сообщений на части (сегменты) с последующей маркировкой каждого сегмента согласно приложению, которое его создало, и приложению, которому оно предназначается.

Как происходит отправка сегментированных сообщений?

Чтобы понять, как происходит процесс отправки следует учесть, что приложений, использующих сеть на устройстве, больше, чем доступных интерфейсов (каналов) обмена данными. Стандартный компьютер имеет всего один сетевой кабельный интерфейс; мобильный телефон – до 4 беспроводных интерфейсов (2 SIM-карты, 1 интерфейс Bluetooth, 1 интерфейс Wi-Fi). Количество приложений иногда исчисляется десятками. Поэтому отправка происходит, как показано на рисунке 3.1.7.1, то есть по мере осуществления сегментации и передачи сегментов для отправки на сетевой интерфейс. По одному каналу могут одновременно передаваться сегментированные сообщения от различных приложений.

Отправка и прием сообщений от нескольких приложений на одном интерфейсе называется *мультиплексированием*.

В нашем примере пользователь осуществляет поиск в Интернете через браузер и одновременно отправляет текстовую информацию в приложении для обмена мгновенными сообщениями. Сегментированные данные от обоих приложений отправляются и принимаются через один проводной интерфейс.

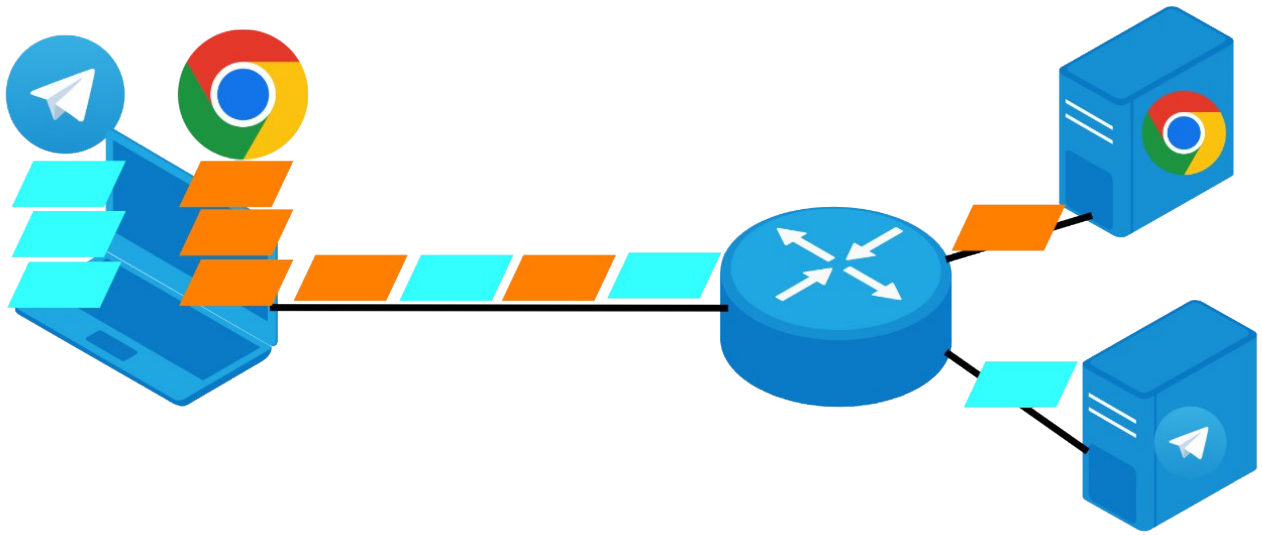


Рисунок 3.1.7.1. – Сегментация и мультиплексирование

3.2. Сетевые модели

Сеть представляет собой сложную систему, в которой пользователи, устройства и программное обеспечение взаимодействуют при помощи некоторых правил – протоколов. В этом разделе будет рассмотрено, как осуществляется такое взаимодействие.

3.2.1. Наборы протоколов

Рассмотрим для начала такой вопрос: для чего необходимо более одного протокола? Дело в том, что для пользователя самое главное – отправить и получить сообщение. Однако, чтобы «достучаться» до собеседника и получить от него ответ с помощью только одного протокола, требовалось бы соблюсти ряд условий:

1) Отправитель и получатель пользуются всего лишь одним приложением;

2) Отправитель и получатель пользуются устройствами одного производителя;
3) Между отправителем и получателем существует всего лишь один канал связи (например, один кабель) без промежуточных устройств.

Теоретически представить себе такую ситуацию возможно, но это будет уже не сеть, и уж тем более не тот Интернет, к которому мы так привыкли. Справедливости ради, нужно упомянуть, что попытки обойтись одним протоколом были: в качестве универсального разрабатывали протокол, который сейчас известен как TCP (*Transmission Control Protocol* – протокол управления пересылкой), но от этой идеи отказались.

Итак, необходимо множество протоколов, имеющих относительно узкую специализацию по функционалу. Проще говоря, легче разработать один протокол под отдельную функцию, чем создавать нечто всеобъемлющее.

Большое количество специализированных протоколов порождает другую проблему: как их координировать между собой, чтобы они взаимодействовали корректно и эффективно?

Для этого необходимо понимать, что устройства при передаче данных оперируют наборами протоколов (их еще называют «стеками протоколов», от английского *stack* – *штабель*), при этом такие наборы могут меняться в зависимости от цели взаимодействия с сетью. Одни протоколы из такого набора будут «работать» непосредственно с приложением, другие будут использоваться для поиска адреса получателя, а некоторые – для отправки сигналов от устройства в сеть и для получения их из сети.

В нашем курсе рассматриваются два основных стека (или модели) протоколов – Open System Interconnect (OSI) Международной организации по стандартизации (ISO) и TCP/IP, последний используется в основном в применении к Интернет-приложениям.

3.2.2. Многоуровневая модель OSI

OSI (Open Systems Interconnection) – концептуальная модель взаимодействия информационных систем, разделенная на семь уровней (по-английски они называются *layer*), нумеруемых снизу вверх (см. рисунок 3.2.2.1).

Для удобства понимания есть простое правило: чем выше уровень, тем ближе к человеку (пользователю), а чем ниже, – тем ближе к среде (например, кабелю).

Взаимодействие пользователя и устройства происходит на *уровне приложений* (L7). Именно здесь формируются сообщения, которые пользователь хочет передать получателю или с помощью которых хочет запросить нужную информацию в сети. Например, когда пользователь набирает в поисковой строке браузера слово «сеть», он формирует запрос к поисковому серверу, например, Google или Yandex.

Далее приложение «решает», передавать данные по сети в открытом виде или в зашифрованном. Протокол передачи гипертекста, HTTP (именно он передает в браузер все веб-страницы после клика на гиперссылку), не шифрует передаваемые данные, а протокол HTTPS (HTTP Secure) – шифрует. Шифрование передаваемой информации происходит на уровне L6 – *представления* данных (англ. *Presentation Layer*).

Рисунок 3.2.2.1. – Модель OSI

Еще ниже, на уровне L5, *приложение* принимает решение, нужно ли устанавливать сеанс связи с получателем для передачи именно этого сообщения? Термин *«сеанс»* (англ. *session*) означает, что два устройства (отправитель и получатель) пересылают друг другу одно большое сообщение по частям, при этом получатель подтверждает получение каждой части, а в конце выдает подтверждение на получение всего сообщения. Сеанс устанавливает гарантию доставки сообщения во-первых полностью, во-вторых - в нужном порядке.

Рассмотрим в качестве примера приложение Telegram. Вероятно, будет неприятно получить сообщение, в котором отсутствует часть текста или перепутаны буквы? То же самое можно сказать о фотографиях и других важных для пользователя данных, где сохранение целостности может быть более приоритетным, чем скорость передачи. Однако, когда дело касается голосовых или видео звонков, на первый план выходит минимизация времени передачи информации, даже если это означает потерю небольшой части переданных данных (небольшая помеха в трубке или легкое искажение на экране получатель простит с бóльшей вероятностью, чем долгие паузы). Именно поэтому при передаче голоса, потокового видео и связанных с ними услуг, установка сеанса связи не требуется.

Далее данные, отформатированные для передачи, с отметкой о необходимости установления сеанса, передаются на *транспортный* уровень – L4. Здесь функционируют протоколы, которые отвечают за сегментацию данных, размер сегментов и скорость обмена сегментами. Сообщение делится на равные отрезки (обычно по 1500 байт), и к каждому добавляется заголовок транспортного уровня, в котором указывается *порт источника*, *порт назначения* и дополнительная информация.

Сегмент – отрезок данных с заголовком транспортного уровня L4.

Чтобы процесс был более наглядным, представим следующую ситуацию: взяли книгу, скопировали каждую страницу на лист А4, затем каждую скопированную страницу положили в отдельный конверт, на конверте подписали название книги, номер страницы

и общее количество страниц. Сегментация произведена. В каждом «сегменте» есть следующие элементы:

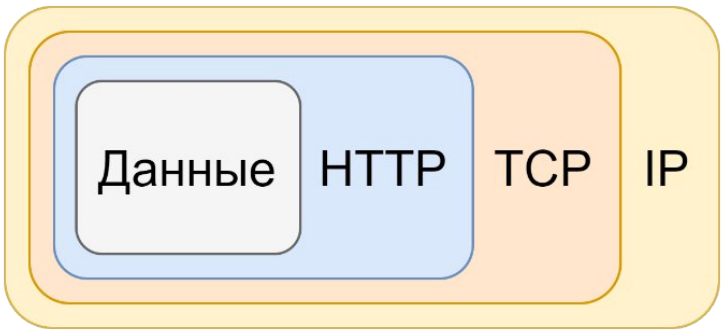
- Текст – данные: информация уровня L7.
- Шрифт, размер, формат: информация уровня L6.
- Сеанс – необходим, так как страницы должны быть собраны в нужном порядке: информация уровня L5.
- Конверт: информация уровня L4 (название книги – приложение, номер страницы и число страниц – номер передаваемого сегмента и общее количество сегментов).

А отрезок данных с заголовком транспортного уровня (в нашем случае протокола TCP) теперь называется *сегмент*.

Сегмент

Рисунок 3.2.2.2. – Состав сегмента транспортного уровня L4

Можно ли теперь пересылать сегмент? Нет, так как пока непонятно куда. Нам нужен *адрес* устройства-получателя в сети. В сетевых технологиях такой адрес называется IP-адресом (по названию протокола *IP – Internet Protocol*) и присваивается он на уровне L3 (сетевом). *Протоколы сетевого уровня занимаются определением способов передачи данных между отправителем и получателем*. Присваивается IP-адрес аналогично тому, как до этого поступили с данными в сегменте: каждый сегмент получает заголовок уровня L3, основной информацией которого служат IP-адрес отправителя (свой адрес отправитель, конечно, знает) и IP-адрес получателя. По нашей аналогии с пересылкой книги происходит следующее: каждый конверт-сегмент теперь кладут в еще один конверт побольше, на котором заполняют город, улицу, номер дома получателя и такие же данные отправителя. Теперь все это называется *пакет*.



Пакет

Рисунок 3.2.2.3. – Состав пакета сетевого уровня L3

***Пакет** – единица данных третьего уровня модели OSI, состоящая из сегмента и заголовка сетевого уровня.*

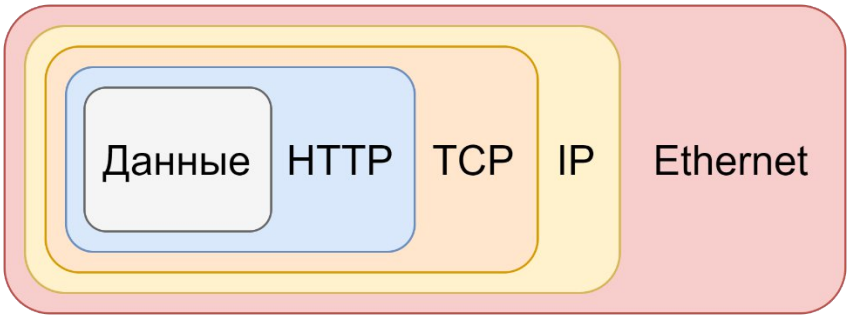
Можно ли теперь пересылать пакет? Тоже нет, так как непонятно, на какое именно

устройство он будет отправлен и через какую *среду* передан. Среда важна, так как для каждой среды на устройстве предусмотрена своя сетевая карта (или адаптер), у нее есть свой аппаратный адрес, который называется MAC-адресом (*MAC* от английского *Media Access Code* – код доступа в среду). MAC-адрес получателя и отправителя (правильнее – источника и назначения) указывается на уровне L2 в специальном заголовке. Пакет с заголовком L2 называется теперь *кадр* (англ. *frame*).

Кадр *– единица данных второго уровня модели OSI, состоящая из пакета и заголовка канального уровня.*

«*Данные», «сегменты», «пакеты» и «кадры»* – это протокольные блоки данных определенных уровней *(PDU – Protocol Data Unit)*.

Как уже говорилось, заголовок L2 зависит от среды, в которую будет отправлен кадр: это может быть протокол Ethernet, если он отправляется по кабелю, или Wi-Fi, если «по воздуху». Закончим нашу аналогию с пересылкой книги: теперь мы помещаем пакет в самый большой конверт из всех возможных и относим его на почту, где ставят индекс нашего почтового отделения и индекс почтового отделения получателя (почтовая служба сама решит, как отправить наше письмо: самолетом, грузовиком или поездом, другими словами, в какой среде осуществить физическую передачу данных).



Кадр

Рисунок 3.2.2.4. – Состав кадра канального уровня L2

Что же происходит на первом - физическом - уровне? На первом уровне происходит физическая передача информации, заключенной в кадре и переведенной в двоичный код (нули и единицы) в виде бит с помощью электрических колебаний, радиоволн или световых сигналов непосредственно в физической среде. Иными словами, на физическом уровне осуществляется преобразование кадров в биты с целью последующей передачи данных в среду передачи. Это объясняет название данного уровня - физический.

Мы рассмотрели отправку сообщения начиная с уровня L7 до уровня L1. При передаче данных с уровня на уровень к ним добавляются заголовки нижестоящих уровней. При этом протокольный блок более высокого уровня как бы помещается – инкапсулируется – в протокольный блок более низкого уровня.

Инкапсуляция *(*англ. *encapsulation)* – процесс формирования новых протокольных блоков более низких уровней из блоков более высоких уровней путем добавления соответствующих заголовков.

При приеме сообщения получателем все процессы повторяются в обратной последовательности. Если кратко представить это на схеме, получится следующее:

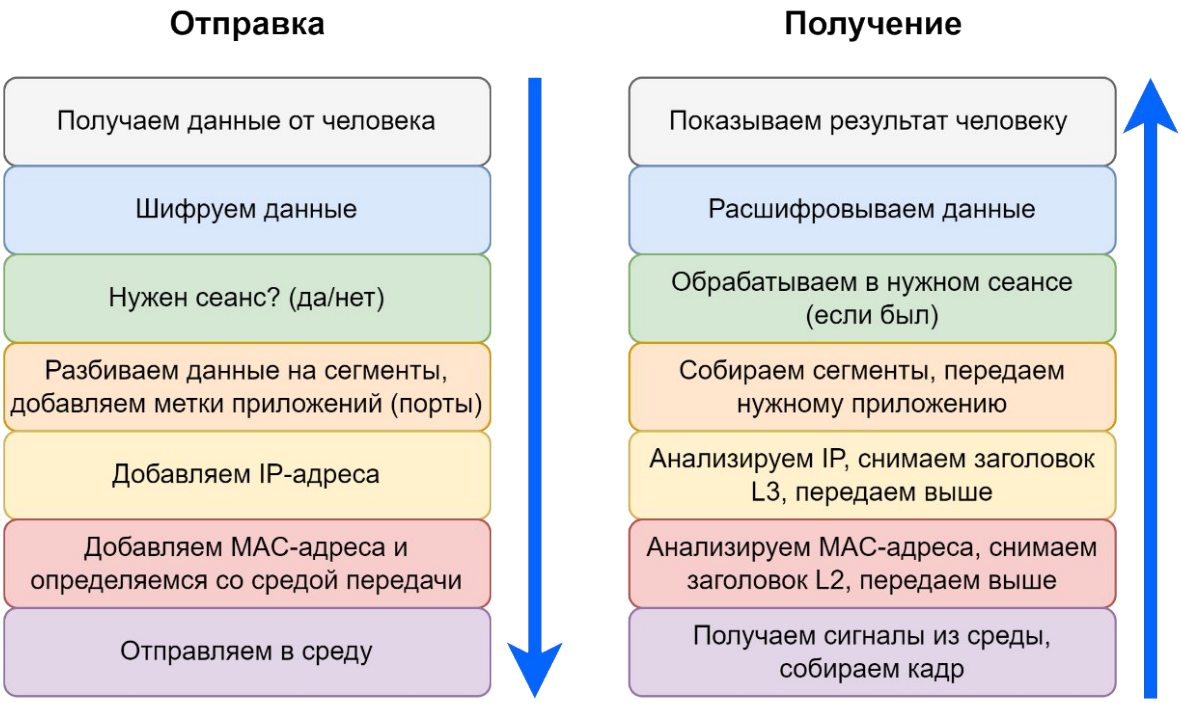


Рисунок 3.2.2.5. – Поэтапный процесс передачи и приема данных

Как видно на рисунке 3.2.2.5, при получении данных протокол более низкого уровня удаляет собственный заголовок и передает полученную информацию протоколу более высокого уровня. Происходит «распаковка» протокольных блоков более высокого уровня, состоящих из протокольных блоков более низкого уровня, или *декапсуляция*.

Декапсуляция *(англ. decapsulation)* – процесс восстановления блоков данных протоколов более высокого уровня путем удаления заголовков протоколов более низкого уровня модели OSI.

3.2.3. Стек TCP/IP

В отличие от модели OSI, которая универсальна и может иметь как практическое, так и образовательное применение, модель (или набор протоколов – стек) TCP/IP более ориентирована на практическое применение в сетях, непосредственно подключенных к сети Интернет. Свое название этот набор протоколов получил от двух главных протоколов Интернета: TCP *(Transmission Control Protocol)* и IP *(Internet Protocol)*. Исторически TCP разрабатывался как универсальный протокол передачи информации между компьютерами в сети ARPANET, которая была предшественницей современного Интернета, но развитие сетевых технологий, постоянно увеличивающееся количество подключаемых устройств различных производителей и новые сервисы привели к тому, что сперва был выделен отдельный протокол IP (он отвечает за адресацию и негарантированную доставку), а затем пришлось налаживать совместимость с другими протоколами.

Стек TCP/IP выделяет всего четыре уровня *(layer)* работы протоколов:

1) Уровень сетевого доступа *(Network Interfaces Layer)*;
2) Интернет-уровень *(Internet Layer)*;
3) Транспортный уровень *(Transport Layer)*;
4) Уровень приложений *(Applications Layer)*.

В стеке TCP/IP меньше уровней по сравнению с моделью OSI, причина этого рассмотрена ниже.

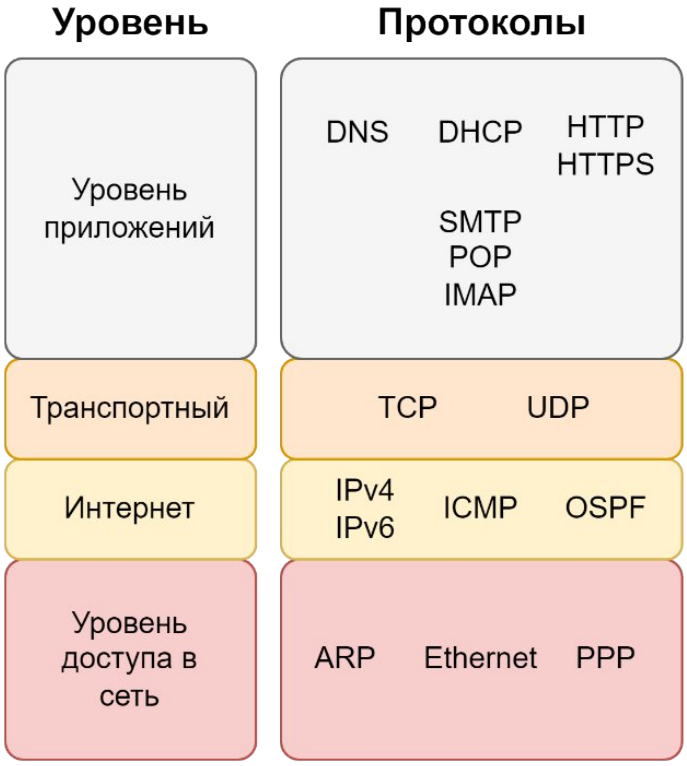


Рисунок 3.2.3.1. – Примеры протоколов в стеке TCP/IP

Основными протоколами стека TCP/IP являются:

На уровне приложений:

- DNS – протокол распознавания доменных имен. Сообщает запрашивающему IP-адрес сайта по его имени;
- DHCP – протокол динамической настройки узла. Позволяет оконечному устройству в автоматическом режиме получить IP-адрес, а также дополнительные параметры, необходимые для работы в сети;
- HTTP/HTTPS – протоколы передачи гипертекста (нешифрованный и шифрованный соответственно). Передают запросы пользователя Web-серверу и содержимое веб-страниц от сервера в браузер пользователя;
- SMTP – простой протокол передачи почты. Отправляет исходящее сообщение на заданный адрес электронной почты и производит подтверждение успешной доставки;
- POP – протокол получения электронной почты. Очищает содержимое «почтового ящика» клиента на сервере после загрузки «писем» пользователем на его локальное устройство;
- IMAP – протокол получения электронной почты. В отличие от POP хранит копии сообщений на сервере после загрузки «писем» пользователем.

На транспортном уровне:

- TCP – протокол, необходимый для надежной доставки данных. Используется для передачи текста, изображений, видео- и музыкальных файлов;
- UDP – протокол, используемый для быстрой передачи данных без контроля

доставки. Применяется для передачи потокового видео (например, ролика в YouTube), голоса и т.д.

На сетевом уровне:

- IPv4 – протокол негарантированной доставки IP версии 4;
- IPv6 – протокол негарантированной доставки IP версии 6;
- ICMP – протокол контрольных сообщений, используется для проверки связности на сетевом уровне (команды ***ping*** и ***traceroute***);
- OSPF – протокол динамической маршрутизации.

На уровне доступа в сеть:

- ARP – протокол, используемый для определения MAC-адреса устройства по его IP-адресу;
- Ethernet – протокол передачи данных по медному кабелю (коаксиальному или витой паре) с множественным доступом к среде;
- PPP – протокол передачи данных по медному телефонному кабелю с возможностью аутентификации.

Все эти протоколы будут подробно рассмотрены в дальнейших главах данного курса.

3.2.4. Сравнение моделей OSI и TCP/IP

Стек протоколов TCP/IP выделяет четыре уровня взаимодействия.

Рисунок 3.2.4.1. – Сравнение моделей OSI и TCP/IP

Физический и канальный уровни объединены в уровень сетевого доступа, управляющий устройствами и средствами подключения к сети.

Сетевой (Интернет) уровень обеспечивает выбор оптимального пути от узла к узлу.

Транспортный уровень обеспечивает аналогичный функционал транспортного уровня модели OSI.

Уровень приложений, представления и сеансовый объединены в уровень

приложений и отвечают за представление информации пользователю, ее форматирование и кодирование, а также за поддержание сессий между двумя узлами.

Логика в таком построении следующая: именно приложение отвечает и за обмен данными с пользователем, и за представление данных, и за установление сеанса. Два «нижних» уровня объединены, поскольку протоколы уровня доступа, непосредственно кодирующие сообщения для отправки, часто работают на уровне аппаратного обеспечения сетевых адаптеров, то есть не могут быть разделены.