Лекция 1. Введение

Эволюция компьютерных сетей началась в 50х годах прошлого века. Первые компьютеры были огромными монстрами и занимали целые здания. Предназначались они лишь для небольшого числа избранных пользователей и не были предназначены для интерактивной работы.

Системы пакетной обработки данных строились как правило на базе одного большого (мощного) компьютера – мэйнфрейма. Пользователи делали перфокарты, содержащие данные для программ и непосредственно сами программы. Операторы в вычислительном центре, где находился мэйнфрейм, вводили данные с перфокарт в компьютер, а распечатанные результаты пользователи получали на след. день. Следовательно, одна сделанная ошибка в стопке перфокарт означала задержку на сутки (как минимум).

Интерактивный режим – это когда пользователь оперативно руководит процессом обработки своих данных и, естественно, он был бы гораздо удобнее. Но интересами программистов и инженеров на заре развития компьютеров пренебрегали по причине огромной стоимости вычислительных машин. Во главу угла ставилась именно эффективность работы компьютера.

В связи с удешевлением процессоров в начале 60х годов появился новый способ организации вычислительного процесса, предлагающий большую интерактивность. Так называемый терминальный режим позволял пользователю работать с мэйнфреймом, не замечая задержки в работе, т.е. время реакции машины было мало. Компьютер работал в режиме разделения времени, отдавая каждому терминалу какой-то квант машинного времени. Вычислительная мощность – централизована, ввод-вывод данных – распределен. Пользователь мог получить доступ к общим ресурсам, что создавало иллюзию работы в сети и единоличного владения компьютером. Именно централизованный характер обработки данных отличает терминальный режим доступа от локальной сети. С другой стороны – предприятия не были готовы к созданию сетей, т.к. т.н. «закон Гроша» очень хорошо отражал действительность того времени. Закон Гроша: «производительность компьютера пропорциональна квадрату его стоимости». За одну и ту же сумму выгоднее купить один более мощный компьютер, чем два менее мощных (их суммарная мощность намного меньше, чем у одного за те же деньги).

Создание глобальных вычислительных сетей началось с возникновением потребности доступа к компьютеру с терминалов, удаленных на сотни и тысячи километров. Терминалы соединялись с компьютером с помощью модемов через телефонные линии. Затем стали появляться системы, где вместо терминалов использовались компьютеры. Таким образом, компьютеры получили возможность обмениваться данными в автоматическом режиме, что является базовым механизмом любой вычислительной сети. Стали реализовываться механизмы обмена файлами, синхронизации баз данных, электронной почты.

Таким образом, вопреки популярному мнению, первыми в мире появились именно глобальные вычислительные сети, на которых и были отработаны ставшие стандартами современные концепции построения сетей, такие как многоуровневое построение коммуникационных протоколов, коммутация и маршрутизация пакетов.

В начале 70х произошел технологический прорыв в области производства компонентов – появились БИС (большие интегральные схемы), что привело к созданию мини-компьютеров. Закон Гроша перестал работать, т.к. несколько мини-компьютеров обеспечивали производительность большую, чем у мэйнфрейма, а стоили дешевле.

Предприятия получили возможность покупать компьютеры для задач управления производством, складом и т.д., но тем не менее, работали они автономно.

С ростом потребностей пользователей, появилась необходимость обмена данными между близко расположенными компьютерами. В ответ на эту потребность организации стали соединять компьютеры между собой и разрабатывать программное обеспечение для их взаимодействия. Так появились первые локальные вычислительные сети. На первых порах компьютеры использовали сложные и разнообразные нестандартные устройства сопряжения, каждое со своим способом передачи данных и типом кабелей. Это создавало, например, большое поле для студенческих работ. Большинство курсовых и дипломных работ назывались «Устройство сопряжения……».

В середине 80х годов утвердились стандартные технологии объединения компьютеров в сеть (Ethernet, Token Ring, Arcnet….). Толчком для развития этих стандартов стало массовое появление

персональных компьютеров, которые стали идеальными элементами для построения сетей, т.к. были достаточно мощными для работы сетевого ПО, а с другой стороны нуждались в объединении для решения сложных вычислительных задач, и, что немаловажно, для разделения доступа к периферийным устройствам, которые били достаточно дорогими (принтер, графопостроитель) либо дисковых массивов хранения данных. Можно сказать, что на этом закончилась эра массового применения мэйнфреймов и терминалов.

В дальнейшем, развитие локальных сетей шло более экстенсивно. Увеличивались скорости сетей. Модифицировались протоколы. С появлением большого количества непрофессиональных пользователей развивались методы прозрачной работы в сети и доступа к разделяемым ресурсам.

Глобальные сети развивались и развиваются немного по другому пути. Скорости даже в 10 Мбит/с считаются во многих сетях непозволительной роскошью. Приходится пользоваться существующими низкоскоростными каналами связи, например, телефонными линиями. Поэтому, основным критерием часто является экономное расходование ресурсов канала связи. Поэтому, процедуры прозрачного доступа долгое время считались непозволительной роскошью.

Что изменилось в наши дни:

- Появляются высокоскоростные территориальные линии связи, что нивелирует разницу между локальными и глобальными сетями. Появляются множество удобных и прозрачных служб доступа к ресурсам (Internet).
- Локальные сети. Вместо соединяющего компьютеры пассивного кабеля в большом количестве появляется коммуникационное оборудование – коммутаторы, маршрутизаторы, шлюзы. Благодаря им, строятся большие корпоративные сети, насчитывающие тысячи компьютеров и имеющие сложную топологию.
- Сами компьютеры становятся крупнее (мощнее). После эйфории от небольших персональных компьютеров, возродился интерес к мощным серверам-мэйнфреймам. Оказывается, их намного проще обслуживать, нем сотни PC, объединенных в сеть.
- Очень важно. Появилась тенденция, несвойственная раньше компьютерам. Возникли новые виды трафика – голос, видео…, что требует внесения изменений в известные ранее протоколы и разработку новых. Этот вид информации чувствителен к задержкам при передаче (рассинхронизация и в конечном итоге искажение информации). Это называется трафик реального времени (в отличие от например, передачи файлов и электронной почты). Эти проблемы решаются различными способами, например, при помощи специально для этого разработанной технологии ATM. В конечном итоге это должно привести к слиянию не только локальных и глобальных сетей, но и информационных сетей (телефония, телевидение). Это должно случиться с массовым переходом от методов коммутации каналов (используемых в телефонии) к методу коммутации пакетов. Наш курс будет посвящен, в том числе и изучению этих методов и соответствующих протоколов.

Существует много типов распределенных вычислительных систем. Их основным признаком является наличие нескольких центров обработки данных.

1.Мультипроцессорные компьютеры. Общая операционная система, распределяющая вычислительную нагрузку между процессорами. Взаимодействие – через общую оперативную память (самый простой метод).

2.Многомашинная система – несколько компьютеров + программные и аппаратные средства. Эффективность снижается, если связь по данным в распараллеливаемых задачах сильная. Обмен – через общие многовходовые периферийные устройства (дисковые массивы).

3.Вычислительные сети – еще большая автономность обрабатываемых данных и слабее программно-аппаратные связи. Связь – сетевые адаптеры и стандартные протоколы. Протяженность линий связи – большая. Основная цель создания ЛВС – разделение локальных ресурсов каждого пользователя сети между всеми.

Весь комплекс программно – аппаратных средств сети может быть описан многослойной моделью. В основе – аппаратный слой стандартизированных компьютерных платформ (от PC до супер ЭВМ, в соответствии с решаемыми задачами).

Второй слой это коммуникационное оборудование. Играет не менее важную роль. Включает мосты, повторители, коммутаторы, маршрутизаторы. Превращаются в основные наряду с

компьютерами как по значимости и влиянию на характеристики сети, так и по цене. Могут иметь свое программное обеспечение и нуждаться в администрировании (напр., маршрутизаторы Cisco).

Третий слой, образующий программную платформу сети – ОС. От того, какие принципы сетевой организации положены в основу ОС, зависит общее функционирование сети (насколько она обеспечивает безопасность и защищенность данных, число пользователей, переносимость, масштабируемость и т.д.).

Самый верхний слой – различные сетевые приложения (БД, почтовые системы, системы автоматизации коллективной работы…). Здесь важно знать предоставляемый программой возможности, а также совместимость с другими сетевыми приложениями и ОС.

Преимущества использования сетей.

- Способность выполнять параллельные вычисления. Концептуальное преимущество. За счет этого может быть достигнута производительность любого сколь угодно мощного компьютера.
- Лучшее соотношение производительность-стоимость.
- Более высокая отказоустойчивость. Отказоустойчивость – это способность системы выполнять свои функции при отказе отдельных элементов аппаратуры и неполной доступности данных. Основа отказоустойчивости – избыточность. При выходе из строя узла, приписанные ему задачи переназначаются. Для этого используются процедуры динамической или статической реконфигурации.
- Адаптация к территориально распределенному характеру прикладных задач. Например, при автоматизации производства, на некоторой территории есть сотрудники, отделы и подразделения, автономно решающие свои задачи и поэтому рациональнее предоставлять им собственные вычислительные средства, но в тоже время, т.к. их задачи взаимосвязаны в рамках общего производственного процесса, вычислительные средства нужно объединять в одну систему. Как раз в такой ситуации адекватное решение – вычислительная сеть.
- Возможность совместного использования данных и устройств. Речь идет об относительно дорогостоящих периферийных устройствах (дисковые массивы, принтеры, графопостроители…).
- Еще один побудительный мотив к построению сетей (гораздо более важный, чем экономия средств за счет разделения дорогой аппаратуры) – обеспечение оперативного доступа к корпоративной информации. В условиях современной жесткой конкуренции очень важна, например, качественная поддержка клиента. А при большой номенклатуре выпускаемых изделий, либо предоставляемых услуг оперативный доступ к справочной базе и надежные связи в корпоративной сети – это залог хорошей работы (пример, мобильная компания).

Лекция 2. Структуризация как средство построения больших сетей.

В сетях с небольшим количеством компьютеров (10…30) обычно применяется одна из типовых топологий – общая шина, кольцо, звезда, т.е. однородные топологии (когда все компьютеры в сети имеют одинаковые права по отношению друг к другу). Какие в этом плюсы:

- Простота наращивания числа компьютеров.
- Простое обслуживание и эксплуатация сети.

При построении больших сетей однородные топологии превращаются из преимуществ в недостатки:

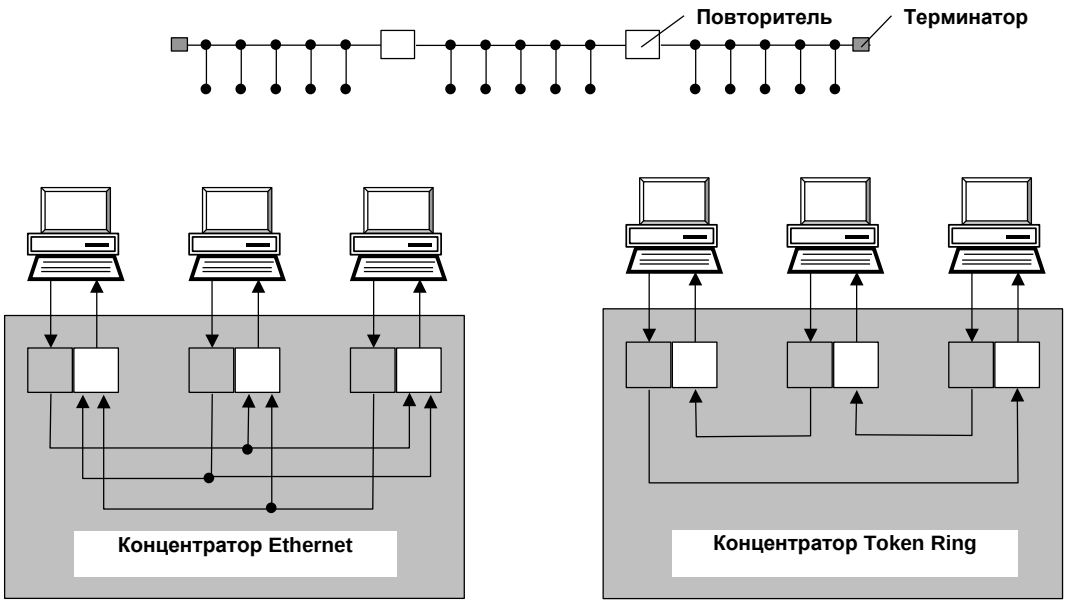
- Ограничения на длину связей между узлами.
- Ограничения на количество узлов в сети.
- Ограничения на количество трафика, порождаемого узлом сети.

Например, та же технология Ethernet позволяет использовать коаксиальный кабель длиной не более 185м., к которому можно подключить не более 30 компьютеров. Однако при интенсивном обмене приходится снижать количество компьютеров до 10…20.

Для снятия подобных ограничений используются специальные методы структуризации сети и специальное структурообразующее оборудование, которое еще называют коммуникационным.

Физическая структуризация сети.

Физическая структуризация сети



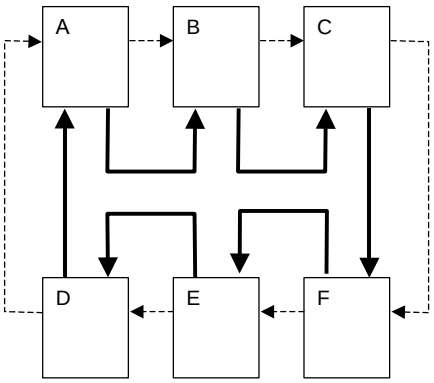
2

Простейшее из коммуникационных устройств – **повторитель**. Используется для увеличения общей длины сети. Преодолеваются ограничения на длину связи за счет улучшения качества передаваемого сигнала (восстановление мощности и амплитуды, улучшение фронтов).

Повторитель, который имеет несколько портов и соединяет несколько физических сегментов, называется **концентратором** или **хабом** (hub – основа, центр деятельности). Концентраторы повторяют сигнал, пришедший по одному из портов, на других портах. Концентраторы характерны почти для всех базовых технологий локальных сетей. Разница в том, на каких именно портах повторяются входные сигналы. Хабы для сетей Ethernet повторяют входной сигнал на всех портах, кроме того, с которого они поступают (слева на слайде). А, например, хабы для сетей Token Ring повторяют входной сигнал только на одном порту – к которому подключен следующий в кольце компьютер (на слайде справа).

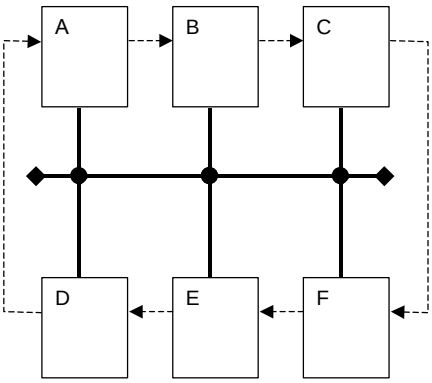
Очень важно: концентратор всегда изменяет физическую топологию сети, но при этом не изменяет логическую топологию.

Физическая структуризация сети



Физическое кольцо

Логическое кольцо



Физическая общая шина

Логическое кольцо

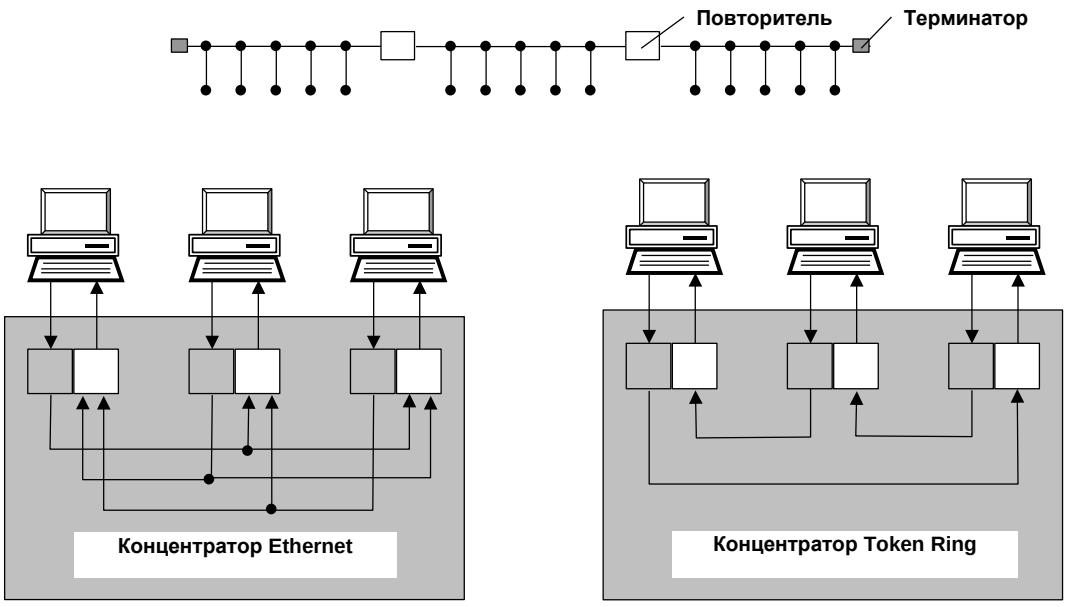
з

Физическая топология – конфигурация связей, образованных кабелем, **логическая топология** – конфигурация информационных потоков между компьютерами.

Во многих случаях физическая и логическая топологии совпадают. Например, сеть, представленная на картинке слева, имеет физическую топологию кольцо. Компьютеры получают доступ к сети за счет передачи друг другу специального кадра – маркера, причем маркер передается от компьютера к компьютеру в том же, порядке, в каком компьютеры образуют физическое кольцо.

Сеть, изображенная справа – пример несовпадения топологий. Фактически компьютеры соединены по технологии «общая шина», но доступ к шине происходит не по алгоритму случайного доступа, а путем передачи маркера в кольцевом порядке: от комп. А – комп. В – затем комп. С и т.д. Здесь порядок передачи маркера определяется не физическими связями в сети, а логическим конфигурированием драйверов сетевых адаптеров. Ничто не мешает настроить драйверы иначе, при этом физическая топология не изменится.

Физическая структуризация сети

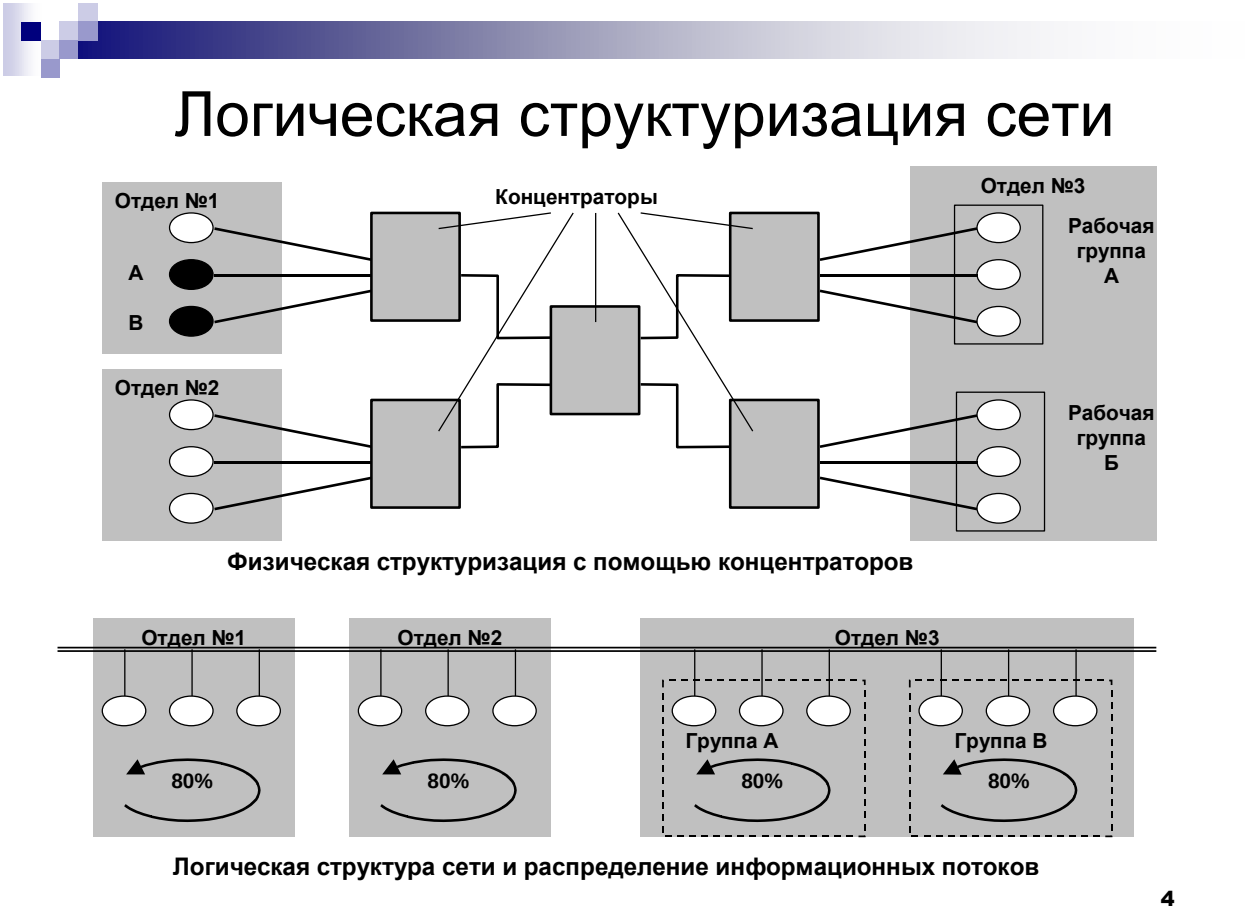


2

Другой пример несовпадения – сеть на рисунке слева. Хаб образует физическую топологию звезда, однако логическая топология осталась без изменения – это общая шина. Логика доступа к сети не меняется – определение занятости среды, получение доступа, обработка коллизий – все остается в силе.

Физическая структуризация с помощью хабов полезна не только для увеличения расстояния между узлами, но и для повышения надежности сети. Например, если концентратор обнаруживает, что узел долго монопольно использует порт – просто отключит его.

Логическая структуризация сети.



Сеть, в которой физические сегменты рассматриваются в качестве одной разделяемой среды, оказывается неадекватна структуре информационных потоков. Например, в сети с общей шиной взаимодействие любой пары компьютеров занимает ее на все время обмена, поэтому при увеличении числа узлов сети шина становится узким местом. Этот случай иллюстрирует верхний рисунок. Например, компьютер А, находясь в одной подсети с компьютером В, посылает ему данные, хабы распространяют их по всем сегментам и, хотя они и не нужны отделам 2 и 3, они вынуждены простаивать, принимая эти данные тоже.

Такая ситуация возникает из за того, что логическая структура сети осталась однородной и решение проблемы состоит состоит в логической структуризации сети.

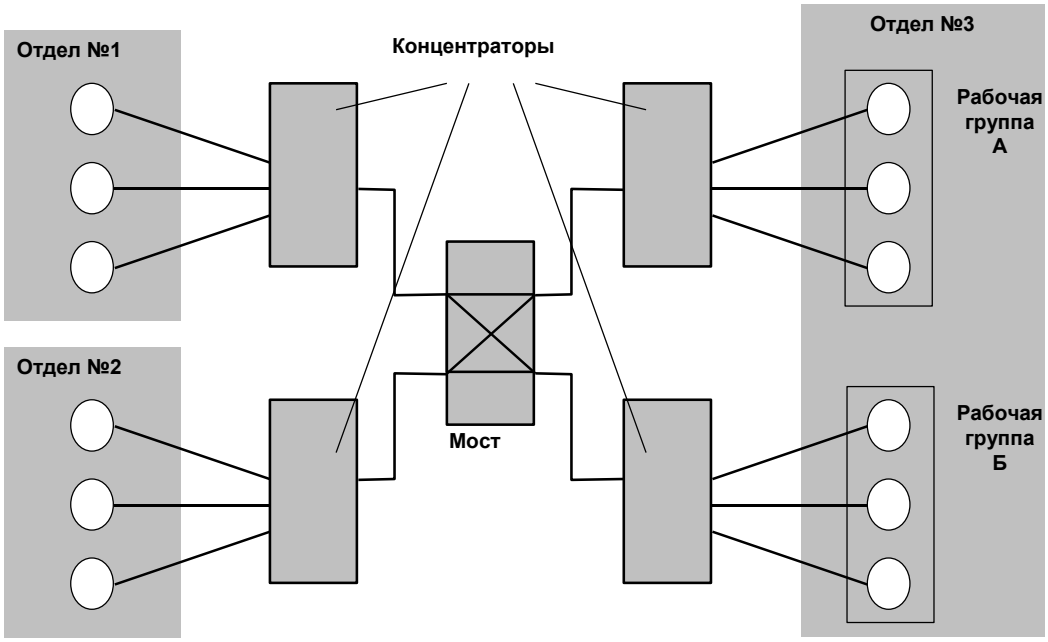
Логическая структуризация сети – это процесс разбиения сети на сегменты с локализованным трафиком.

В рассмотренном выше примере нужно сделать так, чтобы кадры, передаваемые компьютерами отдела 1, не выходили за рамки подсети этого отдела, а в сети других отделов попадали только кадры, адресованные им.

Существует эмпирический закон, в соответствии с которым в логическом сегменте 80% трафика является внутренним, и 20% - внешним.

Для логической структуризации сети используют мосты, коммутаторы, маршрутизаторы и шлюзы.

Логическая структуризация сети



Логическая структуризация сети с помощью моста

5

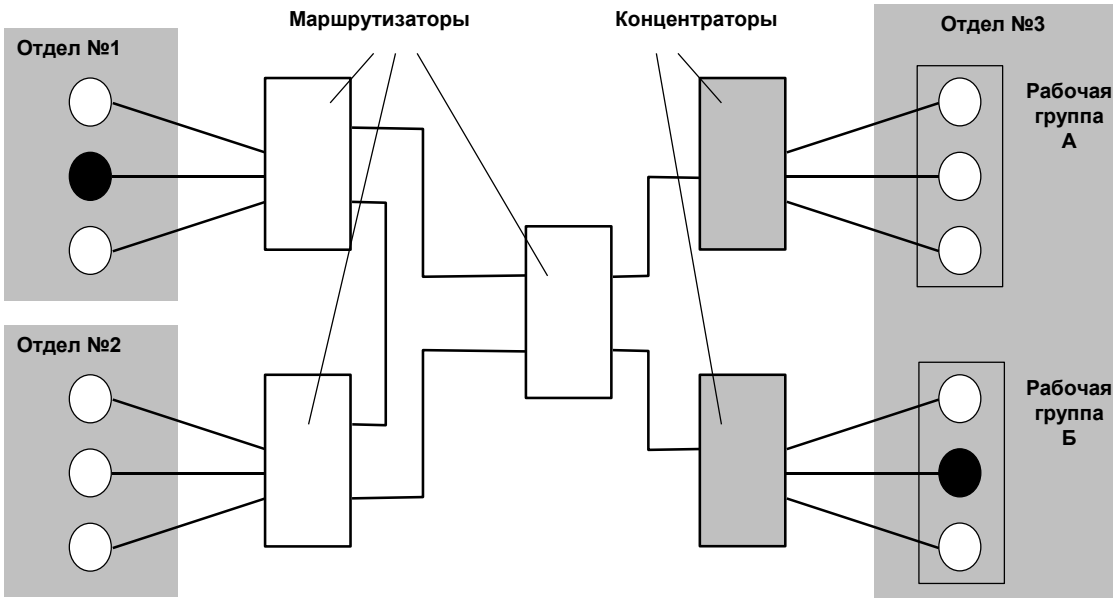
Мост делит разделяемую среду передачи на части (логические сегменты). Фильтрация происходит по аппаратным адресам адаптеров.

На слайде показана сеть, полученная из сети с центральным хабом путем замены его на мост. Точной топологии связей мост не знает и поэтому мост достаточно примитивно представляет связи между узлами – он запоминает на какой порт поступил кадр данных от каждого компьютера и в дальнейшем передает кадры, предназначенные для этого компьютера, только ему в этот порт. Из-за этого применение мостов приводит к ограничению: сегменты должны быть соединены так, чтобы не образовывались замкнутые контуры.

Коммутатор или свитч ничем не отличается от моста по принципу обработки кадров. Различие в том, что каждый его порт оснащен процессором, работающим независимо от других, за счет чего повышается производительность. Можно сказать, что коммутатор – это мост, работающий в параллельном режиме.

Ограничения по топологии связей в сетях с мостами и коммутаторами привели к тому, что появились более эффективные коммуникационные устройства – **маршрутизаторы** (по англ. - **router**).

Логическая структуризация сети



Логическая структуризация сети с помощью маршрутизаторов

6

Маршрутизаторы образуют логические сегменты посредством явной адресации, используя не плоские аппаратные адреса, а составные адреса – IP, где имеются поля номера сети. Все компьютеры, у которых значение этого поля одинаково, принадлежат к одному сегменты, который в этом случае называется подсетью. Маршрутизаторы могут работать в сетях со сколь угодно сложной топологией и в т.ч. с замкнутыми контурами, при этом ои осуществляют выбор наиболее рационального маршрута из нескольких возможных. Еще одна важная особенность маршрутизаторов – они могут связывать в единую сеть подсети, построенные с использованием разных технологий, например, Ethernet и X.25.

Кроме перечисленных устройств, отдельные части сети может соединять **шлюз (gateway)**. Они нужны не для локализации трафика, для объединения сетей с разными типами системного и прикладного программного обеспечения.

Многоуровневый подход. Протокол и интерфейс.

Лирическое отступление.

Ранее упоминался *многоуровневый подход* («слои» адресации аппаратный, числовой, символьный, уровни сетевого взаимодействия). На самом деле многоуровневый подход – это один из методов решения задачи декомпозиции. **Декомпозиция** – это разбиение одной сложной задачи на несколько более простых задач-модулей. Процедура декомпозиции включает в себя четкое определение функций каждого модуля и интерфейсов между ними. Так вот: многоуровневый подход заключается в следующем. Все множество модулей разбивают на уровни, которые сформированы следующим образом: для выполнения своих задач модуль обращается с запросом только к модулям нижележащего уровня. А результаты работы могут быть переданы только модулю соседнего вышележащего уровня. Набор функций, который нижележащий уровень предоставляет вышележащему – это **интерфейс**.

Чтобы пояснить понятия **протокол-интерфейс** рассмотрим простой пример.

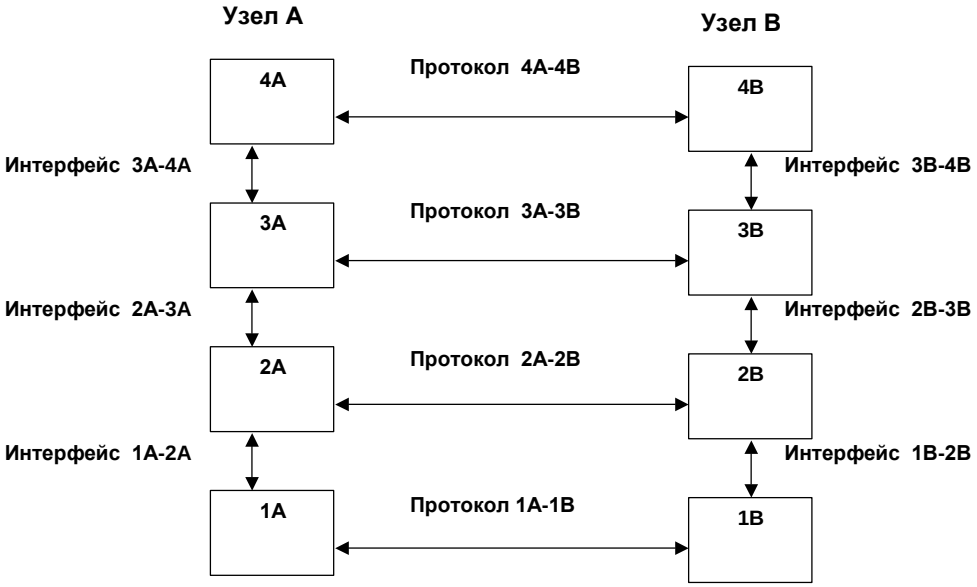
Многоуровневый подход

Пример многоуровневого взаимодействия

7

Есть 2 предприятия А и В, связанные каким-либо деловым сотрудничеством. Между предприятиями существуют многочисленные договоренности и соглашения, например регулярные поставки продукции одного предприятия другому. Начальник предприятия А регулярно (каждый месяц) посылает официальное сообщение начальнику предприятия В о том, сколько и чего доступно на складе. В ответ, начальник В посылает заявку сколько и чего нужно. Это и есть установленный порядок взаимодействия, который называется **протокол**. В данном случае – протокол уровня начальников. Начальники посылают свои заявки не сами, а через своих секретарей. Порядок взаимодействия начальника и секретара –это межуровневый **интерфейс**. Например, на предприятии А обмен идет по электронной почте, а на предприятии В начальник общается с секретарем по телефону. Т.е., интерфейсы начальник-секретарь отличаются. После того, как сообщения переданы секретарям, начальников не волнует, как эти сообщения будут переданы дальше. Это может быть почта, факс, курьер и т.д. Выбор способа передачи – это компетенция секретарей, они решают этот вопрос не уведомляя начальников и связываясь только между собой. Это протокол взаимодействия секретарей. При решении других вопросов начальники могут общаться по другим протоколам-правилам и это не повлияет на работу секретарей, для которых не важно, что отправлять, а важно, чтобы сообщения дошли до адресата. Мы имеем дело с двумя уровнями – уровнем начальников и уровнем секретарей и каждый из них имеет собственный протокол, который может быть изменен независимо от протокола другого уровня. Эта независимость протоколов и есть основное достоинство многоуровневого подхода.

Многоуровневый подход



Взаимодействие двух узлов

8

Иллюстрирует то же самое, но для большего количества уровней иерархии на примере взаимодействия двух узлов сети.

Еще раз определения:

Протокол – формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты одного уровня в разных узлах.

Интерфейс – правила взаимодействия сетевых компонентов соседних уровней одного узла, реализуемые с помощью стандартизованных сообщений.

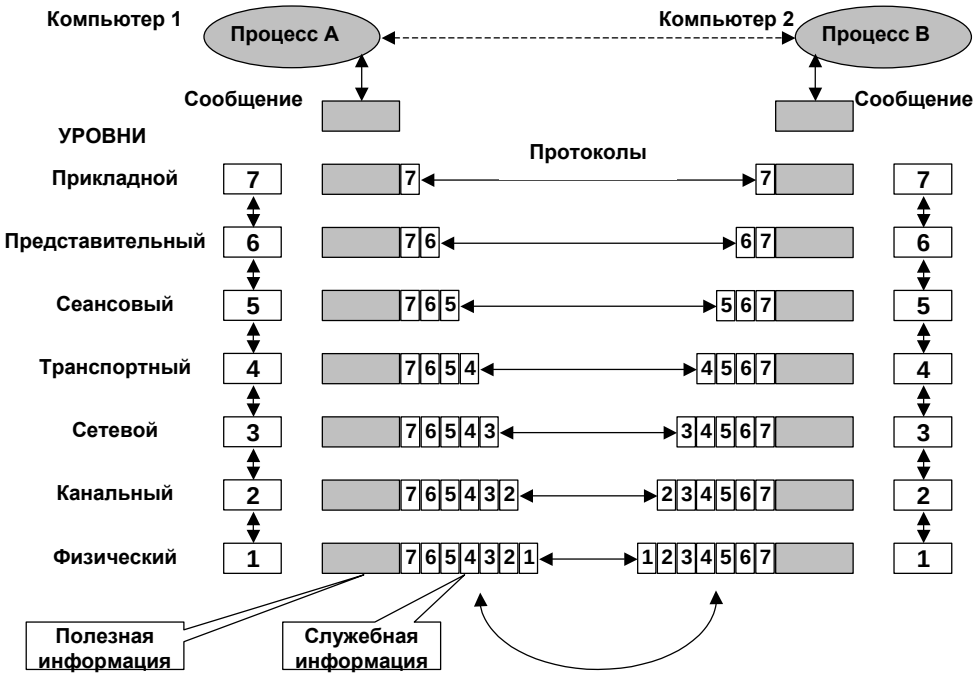
В этом случае программно-аппаратные средства каждого уровня должны обеспечивать собственный протокол и интерфейс с соседними уровнями.

Иерархический набор протоколов, достаточный для организации взаимодействия узлов в сети, называется **стеком коммуникационных протоколов.**

Модель OSI.

В начале 80-х международная организация по стандартизации ISO разработала эталонную модель OSI в качестве модели для архитектуры компьютерных протоколов. Разработчики модели OSI предполагали, что эта модель и протоколы, разрабатываемые в ее рамках, будут доминировать в компьютерной связи и в конце концов вытеснят конкурирующие модели, такие как TCP/IP. Этого не произошло. Хотя в контексте OSI было создано много полезных протоколов, сама модель не получила всеобщего признания. Напротив, доминирующей стала именно TCP/IP. Причина была в том, что на момент разработки OSI аналогичные протоколы TCP/IP были работоспособными и отлаженными. Тем не менее, рассмотрение этой модели важно для понимания общих принципов многоуровневого подхода, к тому-же, различия с TCP/IP в составе и особенностях протоколов – тема для более углубленного курса и в наши планы не входит. Кратко эти различия рассмотрим дальше.

Модель OSI

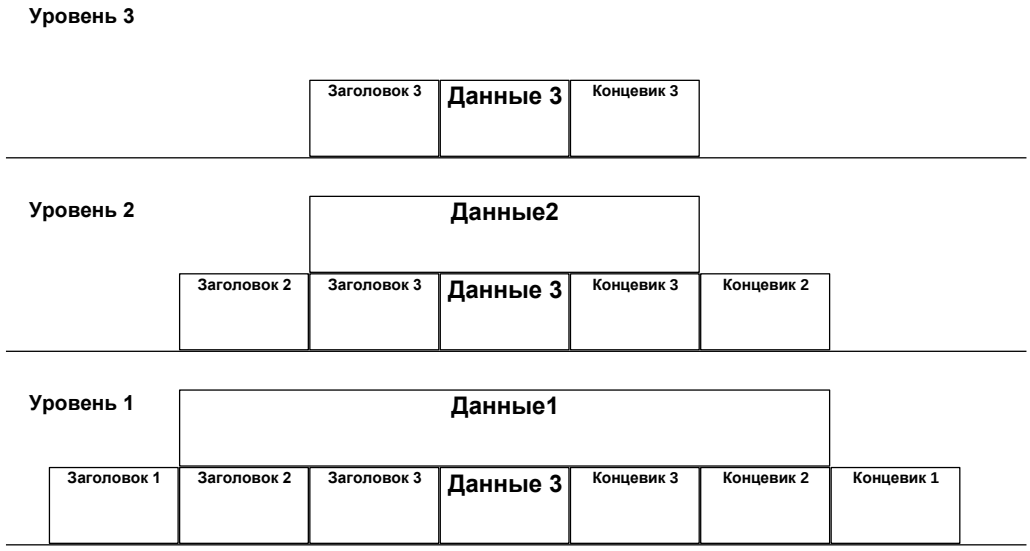


Семиуровневая модель взаимодействия открытых систем 9

Полное описание модели OSI занимает примерно 1000 страниц текста. Вкратце: в модели OSI семь уровней взаимодействия: прикладной, представительный сеансовый, транспортный, сетевой, канальный и физический. Модель не включает средства взаимодействия приложений конечных пользователей, к тому же, приложение может взять на себя функции некоторых верхних уровней модели.

Итак, пусть приложение обращается с запросом к прикладному уровню, например, к файловой службе. На основании этого запроса программное обеспечение прикладного уровня формирует сообщение стандартного формата. Обычно сообщение состоит из заголовка и поля данных. В заголовке – служебная информация, которую нужно передать прикладному уровню адресата, чтобы сообщить ему, какую работу нужно выполнить (например, о месте нахождения файла и о типе операции, которую необходимо над ним выполнить). Поле данных может быть пустым или содержать данные, например, те, которые нужно записать в удаленный файл. Но для того, чтобы доставить эту информацию по назначению, предстоит решить еще много задач, ответственность за которые лежит на нижних уровнях.

Модель OSI



Вложенность сообщений различных уровней

10

После формирования сообщения прикладной уровень направляет его вниз по стеку. Протокол представительного уровня добавляет собственную служебную информацию, в котором содержатся указания для представительного протокола машины-адресата. Получившееся сообщение передается вниз сеансовому уровню, который, в свою очередь, добавляет свой заголовок и т.д. Некоторые протоколы помещают служебную информацию не только в начале сообщения, но и в виде т.н. «концевика». Наконец, сообщение достигает нижнего, физического уровня, который собственно и передает его по линиям связи машине-адресату. К этому моменту сообщение «обрастает» заголовками всех уровней.

Когда сообщение по сети поступает на машину-адресат, оно принимается ее физическим уровнем и последовательно перемещается вверх с уровня на уровень. Каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняет свои функции, удаляет свой заголовок и передает оставшееся сообщение (которое для этого уровня трактуется как поле данных) протоколу наверху.

Уровни модели OSI.

Физический уровень имеет дело с передачей битов по физическим каналам связи, таким, например, как коаксиальный кабель, витая пара, оптоволоконный кабель. К этому уровню имеют отношение характеристики физических сред передачи данных, такие как полоса пропускания, помехозащищенность, волновое сопротивление и другие. На этом же уровне определяются характеристики электрических сигналов, передающих дискретную информацию, например, крутизна фронтов импульсов, уровни напряжения или тока передаваемого сигнала, тип кодирования, скорость передачи сигналов. Кроме этого, здесь стандартизуются типы разъемов и назначение каждого контакта.

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом.

Примером протокола физического уровня может служить спецификация 10Base-T технологии Ethernet, которая определяет в качестве используемого кабеля неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем RJ-45, максимальную длину физического сегмента 100

метров, манчестерский код для представления данных в кабеле, а также некоторые другие характеристики среды и электрических сигналов.

Канальный уровень

На физическом уровне просто пересылаются биты. При этом не учитывается, что в некоторых сетях, в которых линии связи используются (разделяются) попеременно несколькими парами взаимодействующих компьютеров, физическая среда передачи может быть занята. Поэтому одной из задач канального уровня является проверка доступности среды передачи. Другой задачей канального уровня является реализация механизмов обнаружения и коррекции ошибок. Для этого на канальном уровне биты группируются в наборы, называемые **кадрами**. Канальный уровень обеспечивает корректность передачи каждого кадра, помещая специальную последовательность бит в начало и конец каждого кадра, для его выделения, а также вычисляет контрольную сумму, обрабатывая все байты кадра определенным способом и добавляя контрольную сумму к кадру. Когда кадр приходит по сети, получатель снова вычисляет контрольную сумму полученных данных и сравнивает результат с контрольной суммой из кадра. Если они совпадают, кадр считается правильным и принимается. Если же контрольные суммы не совпадают, то фиксируется ошибка. Канальный уровень может не только обнаруживать ошибки, но и исправлять их за счет повторной передачи поврежденных кадров. Необходимо отметить, что функция исправления ошибок не является обязательной для канального уровня, поэтому в некоторых протоколах этого уровня она отсутствует, например, в Ethernet.

В протоколах канального уровня, используемых и локальных сетях, заложена определенная структура связей между компьютерами и способы их адресации. Хотя канальный уровень и обеспечивает доставку кадра между любыми двумя узлами локальной сети, он это делает только в сети с совершенно определенной топологией связей, именно той топологией, для которой он был разработан. К таким типовым топологиям, поддерживаемым протоколами канального уровня локальных сетей, относятся общая шина, кольцо и звезда, а также структуры, полученные из них с помощью мостов и коммутаторов. Примерами протоколов канального уровня являются протоколы Ethernet, Token Ring, FDDI. В локальных сетях протоколы канального уровня используются компьютерами, мостами, коммутаторами и маршрутизаторами. В компьютерах функции канального уровня реализуются совместными усилиями сетевых адаптеров и их драйверов.

В глобальных сетях, которые редко обладают регулярной топологией, канальный уровень часто обеспечивает обмен сообщениями только между двумя соседними компьютерами, соединенными индивидуальной линией связи. Примером протокола «точка-точка» (как часто называют такие протоколы) могут служить широко распространенный протокол PPP. В таких случаях для доставки сообщений между конечными узлами через всю сеть используются средства сетевого уровня. Именно так организованы сети X.25. Иногда в глобальных сетях функции канального уровня в чистом виде выделить трудно, так как в одном и том же протоколе они объединяются с функциями сетевого уровня. Примерами такого подхода могут служить протоколы технологии ATM.

В целом канальный уровень представляет собой весьма мощный и законченный набор функций по пересылке сообщений между узлами сети, В некоторых случаях протоколы канального уровня оказываются самодостаточными транспортными средствами и могут допускать работу поверх них непосредственно протоколов прикладного уровня или приложений, без привлечения средств сетевого и транспортного уровней.

Тем не менее для обеспечения качественной транспортировки сообщений в сетях любых топологий и технологий функций канального уровня часто оказывается недостаточно, поэтому в модели OSI решение этой задачи возлагается на два следующих уровня — сетевой и транспортный.

Сетевой уровень служит для образования единой транспортной системы, объединяющей несколько сетей, причем эти сети могут использовать совершенно различные принципы передачи сообщений между конечными узлами и обладать произвольной структурой связей. Функции сетевого уровня достаточно разнообразны. Начнем их рассмотрение на примере объединения локальных сетей.

Протоколы канального уровня локальных сетей обеспечивают доставку данных между любыми узлами только в сети с соответствующей типовой топологией, например топологией иерархической звезды. Это очень жесткое ограничение, которое не позволяет строить сети с развитой структурой, например, сети, объединяющие несколько сетей предприятия в единую сеть, или высоконадежные сети,

в которых существуют избыточные связи между узлами. Можно было бы усложнять протоколы канального уровня для поддержания петлевидных связей, но принцип разделения обязанностей между уровнями приводит к другому решению. Чтобы с одной стороны сохранить простоту процедур передачи данных для типовых топологий, а с другой допустить использование произвольных топологий, вводится дополнительный сетевой уровень.

На сетевом уровне сам термин сеть наделяют специфическим значением. В данном случае под сетью понимается совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий и использующих для передачи данных один из протоколов канального уровня, определенный для этой топологии.

Внутри сети доставка данных обеспечивается соответствующим канальным уровнем, а вот доставкой данных между сетями занимается сетевой уровень, который и поддерживает возможность правильного выбора маршрута передачи сообщения даже в том случае, когда структура связей между составляющими сетями имеет характер, отличный от принятого в протоколах канального уровня.

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами.

Маршрутизатор — это устройство, которое собирает информацию о топологии межсетевых соединений и на ее основании пересылает пакеты сетевого уровня в сеть назначения. Чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно совершить некоторое количество транзитных передач между сетями, или хопов (от hop — прыжок), каждый раз выбирая подходящий маршрут. Таким образом, маршрут представляет собой последовательность маршрутизаторов, через которые проходит пакет.

Проблема выбора наилучшего пути называется маршрутизацией, и ее решение является одной из главных задач сетевого уровня. Эта проблема осложняется тем, что самый короткий путь не всегда самый лучший. Часто критерием при выборе маршрута является время передачи данных по этому маршруту; оно зависит от пропускной способности каналов связи и интенсивности трафика, которая может изменяться с течением времени. Некоторые алгоритмы маршрутизации пытаются приспособиться к изменению нагрузки, в то время как другие принимают решения на основе средних показателей за длительное время. Выбор маршрута может осуществляться и по другим критериям, например надежности передачи.

Сетевой уровень решает также задачи согласования разных технологий, упрощения адресации в крупных сетях и создания надежных и гибких барьеров на пути нежелательного трафика между сетями.

Сообщения сетевого уровня принято называть пакетами (packets). При организации доставки пакетов на сетевом уровне используется понятие «номер сети». В этом случае адрес получателя состоит из старшей части — номера сети и младшей — номера узла в этой сети. Все узлы одной сети должны иметь одну и ту же старшую часть адреса, поэтому термину «сеть» на сетевом уровне можно дать и другое, более формальное определение: сеть — это совокупность узлов, сетевой адрес которых содержит один и тот же номер сети.

На сетевом уровне определяются два вида протоколов. Первый вид — сетевые протоколы (routed protocols) — реализуют продвижение пакетов через сеть. Именно эти протоколы обычно имеют в виду, когда говорят о протоколах сетевого уровня. Однако часто к сетевому уровню относят и другой вид протоколов, называемых протоколами обмена маршрутной информацией или просто протоколами маршрутизации (routing protocols). С помощью этих протоколов маршрутизаторы собирают информацию о топологии межсетевых соединений. Протоколы сетевого уровня реализуются программными модулями операционной системы, а также программными и аппаратными средствами маршрутизаторов.

На сетевом уровне работают протоколы еще одного типа, которые отвечают за отображение адреса узла, используемого на сетевом уровне, в локальный адрес сети. Такие протоколы часто называют протоколами разрешения адресов — Address Resolution Protocol, ARP. Иногда их относят не к сетевому уровню, а к канальному, хотя тонкости классификации не изменяют их сути.

Примерами протоколов сетевого уровня являются протокол межсетевого взаимодействия IP стека TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell.

Транспортный уровень.

На пути от отправителя к получателю пакеты могут быть искажены или утеряны. Хотя некоторые приложения имеют собственные средства обработки ошибок, существуют и такие, которые предпочитают сразу иметь дело с надежным соединением. Транспортный уровень обеспечивает приложениям или верхним уровням стека — прикладному и сеансовому — передачу данных с той степенью надежности, которая им требуется. Модель OSI определяет пять классов сервиса, предоставляемых транспортным уровнем. Эти виды сервиса отличаются качеством предоставляемых услуг: срочностью, возможностью восстановления прерванной связи, наличием средств мультиплексирования нескольких соединений между различными прикладными протоколами через общий транспортный протокол, а главное — способностью к обнаружению и исправлению ошибок передачи, таких как искажение, потеря и дублирование пакетов.

Выбор класса сервиса транспортного уровня определяется, с одной стороны, тем, в какой степени задача обеспечения надежности решается самими приложениями и протоколами более высоких, чем транспортный, уровней, а с другой стороны, этот выбор зависит от того, насколько надежной является система транспортировки данных в сети, обеспечиваемая уровнями, расположенными ниже транспортного — сетевым, канальным и физическим. Так, например, если качество каналов передачи связи очень высокое и вероятность возникновения ошибок, не обнаруженных протоколами более низких уровней, невелика, то разумно воспользоваться одним из облегченных сервисов транспортного уровня, не обремененных многочисленными проверками, квитированием и другими приемами повышения надежности. Если же транспортные средства нижних уровней изначально очень ненадежны, то целесообразно обратиться к наиболее развитому сервису транспортного уровня, который работает, используя максимум средств для обнаружения и устранения ошибок, — с помощью предварительного установления логического соединения, контроля доставки сообщений по контрольным суммам и циклической нумерации пакетов, установления тайм-аутов доставки и т. п.

Как правило, все протоколы, начиная с транспортного уровня и выше, реализуются программными средствами конечных узлов сети — компонентами их сетевых операционных систем. В качестве примера транспортных протоколов можно привести протоколы TCP и UDP стека TCP/IP и протокол SPX стека Novell.

Протоколы нижних четырех уровней обобщенно называют сетевым транспортом или транспортной подсистемой, так как они полностью решают задачу транспортировки сообщений с заданным уровнем качества в составных сетях с произвольной топологией и различными технологиями. Остальные три верхних уровня решают задачи предоставления прикладных сервисов на основании имеющейся транспортной подсистемы.

Сеансовый уровень обеспечивает управление диалогом: фиксирует, какая из сторон является активной в настоящий момент, предоставляет средства синхронизации. Последние позволяют вставлять контрольные точки в длинные передачи, чтобы в случае отказа можно было вернуться назад к последней контрольной точке, а не начинать все с начала. На практике немногие приложения используют сеансовый уровень, и он редко реализуется в виде отдельных протоколов, хотя функции этого уровня часто объединяют с функциями прикладного уровня и реализуют в одном протоколе.

Представительный уровень имеет дело с формой представления передаваемой по сети информации, не меняя при этом ее содержания. За счет уровня представления информация, передаваемая прикладным уровнем одной системы, всегда понятна прикладному уровню другой системы. С помощью средств данного уровня протоколы прикладных уровней могут преодолеть синтаксические различия в представлении данных или же различия в кодах символов, например кодов ASCII. На этом уровне может выполняться шифрование и дешифрование данных, благодаря которому секретность обмена данными обеспечивается сразу для всех прикладных служб. Примером такого протокола является протокол Secure Socket Layer (SSL), который обеспечивает секретный обмен сообщениями для протоколов прикладного уровня стека TCP/IP.

Прикладной уровень — это в действительности просто набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые Web-страницы, а также организуют свою совместную работу, например,

с помощью протокола электронной почты. Единица данных, которой оперирует прикладной уровень, обычно называется сообщением (message).

Существует очень большое разнообразие служб прикладного уровня. Приведем в качестве примера хотя бы несколько наиболее распространенных реализаций файловых служб: NCP в операционной системе Novell NetWare, SMB в Microsoft Windows NT, NFS, FTP и TFTP, входящие в стек TCP/IP.

Три нижних уровня — физический, канальный и сетевой — являются сетезависимыми, то есть протоколы этих уровней тесно связаны с технической реализацией сети и используемым коммуникационным оборудованием. Например, переход на оборудование FDDI означает полную смену протоколов физического и канального уровней во всех узлах сети.

Три верхних уровня — прикладной, представительный и сеансовый — ориентированы на приложения и мало зависят от технических особенностей построения сети. На протоколы этих уровней не влияют какие бы то ни было изменения в топологии сети, замена оборудования или переход на другую сетевую технологию. Так, переход от Ethernet на более высокоскоростную технологию не потребует никаких изменений в программных средствах, реализующих функции прикладного, представительного и сеансового уровней.

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних. Это позволяет разрабатывать приложения, не зависящие от технических средств непосредственной транспортировки сообщений.

Еще раз, это важно… можно даже записать.

Физический уровень

- обеспечивает передачу неструктурированного потока битов по физическому носителю
- имеет дело с механическими, электрическими, функциональными и процедурными характеристиками доступа к физической линии связи

Канальный уровень или уровень передачи данных

- обеспечивает надежную передачу данных по физической линии
- посылает блоки (кадры) с необходимой синхронизацией, контролем ошибок и управлением потоком

Сетевой уровень

- обеспечивает независимость верхних уровней от технологий передачи данных и коммутации
- отвечает за установку и разрыв соединений и за управление соединениями

Транспортный уровень

- обеспечивает надежный и прозрачный перенос данных между конечными узлами
- обеспечивает сквозное восстановление после ошибок и управление потоком

Сеансовый уровень

- предоставляет структуру управления для взаимодействия приложений
- устанавливает и разрывает сеансы между приложениями

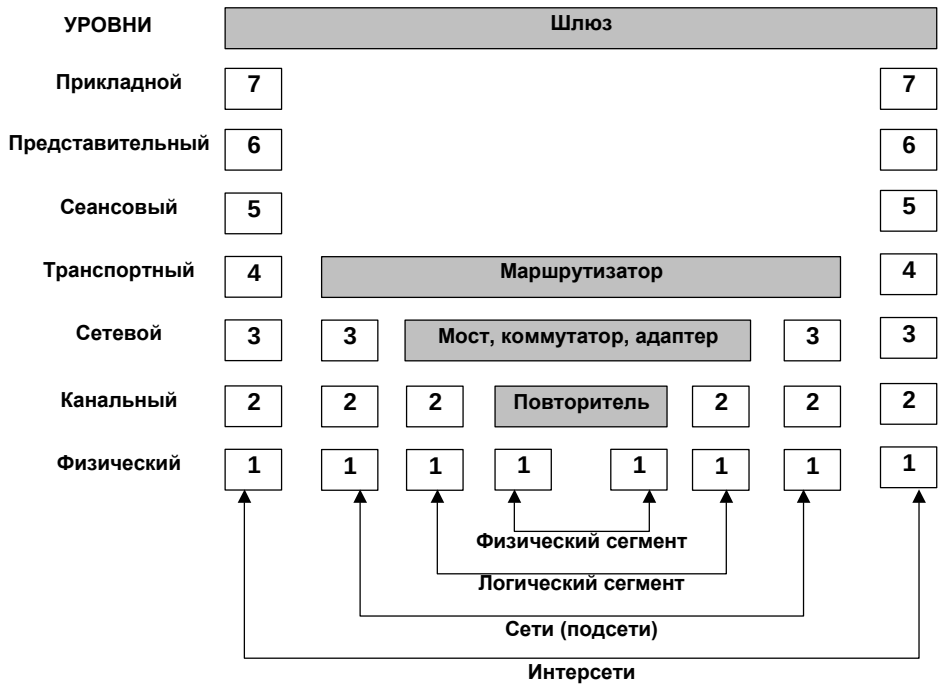
Представительный уровень

- обеспечивает прикладному процессу независимость от синтаксиса представления данных

Прикладной уровень

- обеспечивает доступ пользователей к окружению OSI
- предоставляет распределенные информационные службы

Модель OSI



Соответствие функций коммуникационных устройств уровням модели OSI 11

На слайде показаны уровни модели OSI, на которых работают различные элементы сети. Компьютер с установленной на нем сетевой ОС взаимодействует с другим компьютером с помощью протоколов всех семи уровней. Это взаимодействие компьютеры осуществляют опосредовано через различные коммуникационные устройства: концентраторы, модемы, мосты, коммутаторы, маршрутизаторы, мультиплексоры. В зависимости от типа коммуникационное устройство может работать либо только на физическом уровне (повторитель), либо на физическом и канальном (мост), либо на физическом, канальном и сетевом, иногда захватывая и транспортный уровень (маршрутизатор).

Модель OSI представляет хотя и очень важную, но только одну из многих моделей коммуникаций. Эти модели и связанные с ними стеки протоколов могут отличаться количеством уровней, их функциями, форматами сообщений, службами, поддерживаемыми на верхних уровнях, и прочими параметрами.

Понятие «открытая система»

Модель OSI, как это следует из ее названия (Open System Interconnection), описывает взаимосвязие открытых систем. Что же такое открытая система?

В широком смысле открытой системой может быть названа любая система (компьютер, вычислительная сеть, ОС, программный пакет, другие аппаратные и программные продукты), которая построена в соответствии с открытыми спецификациями.

Спецификация (в вычислительной технике) понимают формализованное описание аппаратных или программных компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, ограничений и особых характеристик. Понятно, что не всякая спецификация является стандартом. В свою очередь, под открытыми спецификациями понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами.

Использование при разработке систем открытых спецификаций позволяет третьим сторонам разрабатывать для этих систем различные аппаратные или программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Модель OSI касается только одного аспекта открытости, а именно открытости средств взаимодействия устройств, связанных в вычислительную сеть. Здесь под открытой системой понимается сетевое устройство, готовое взаимодействовать с другими сетевыми устройствами с использованием стандартных правил, определяющих формат, содержание и значение принимаемых и отправляемых сообщений.

Если две сети построены с соблюдением принципов открытости, то это дает следующие преимущества:

- возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся одного и того же стандарта;
- возможность безболезненной замены отдельных компонентов сети другими, более совершенными, что позволяет сети развиваться с минимальными затратами;
- возможность легкого сопряжения одной сети с другой;
- простота освоения и обслуживания сети.

Ярким примером открытой системы является международная сеть Internet. Эта сеть развивалась в полном соответствии с требованиями, предъявляемыми к открытым системам. В разработке ее стандартов принимали участие тысячи специалистов-пользователей этой сети из различных университетов, научных организаций и фирм-производителей вычислительной аппаратуры и программного обеспечения, работающих в разных странах. Само название стандартов, определяющих работу сети Internet — Request For Comments (RFC), что можно перевести как «запрос на комментарии», — показывает гласный и открытый характер принимаемых стандартов. В результате сеть Internet сумела объединить в себе самое разнообразное оборудование и программное обеспечение огромного числа сетей, разбросанных по всему миру.

Если результаты практических исследований показывают эффективность предлагаемого стандарта, то ему, со всеми внесенными изменениями, присваивается статус проекта стандарта. Затем в течение не менее 4-х месяцев проходят его дальнейшие испытания «на прочность», в число которых входит создание по крайней мере двух программных реализаций.

Если во время пребывания в ранге проекта стандарта в документ не было внесено никаких исправлений, то ему может быть присвоен статус официального стандарта Internet. Список утвержденных официальных стандартов Internet публикуется в виде документа RFC и доступен в Internet.

Следует заметить, что все стандарты Internet носят название RFC с соответствующим порядковым номером, но далеко не все RFC являются стандартами Internet — часто эти документы представляют собой комментарии к какому-либо стандарту или просто описания некоторой проблемы Internet.

Стандартные стеки коммуникационных протоколов

Важнейшим направлением стандартизации в области вычислительных сетей является стандартизация коммуникационных протоколов. В настоящее время в сетях используется большое количество стеков коммуникационных протоколов. Наиболее популярными являются стеки: TCP/IP, IPX/SPX, NetBIOS/SMB и OSI. Все эти стеки на нижних уровнях — физическом и канальном, — используют одни и те же хорошо стандартизованные протоколы Ethernet, Token Ring, FDDI и некоторые другие, которые позволяют использовать во всех сетях одну и ту же аппаратуру. Зато на верхних уровнях все стеки работают по своим собственным протоколам.

Стандартные стеки коммуникационных протоколов

Модель OSI	IBM/Microsoft	TCP/IP	Novell	Стек OSI
Прикладной	SMB	TelNet, FTP, SNMP, SMTP, WWW	NCP, SAP	X.400, X.500, FTAM
Представительный				Представительный протокол OSI
Сеансовый	NetBIOS	TCP		Сеансовый протокол OSI
Транспортный			SPX	Транспортный протокол OSI
Сетевой		IP, RIP, OSPF	IPX, RIP, NLSP	ES-ES IS-IS
Канальный	Ethernet, Token Ring, FDDI, Fast Ethernet, SLIP, 100VG-AnyLAN, X.25, ATM, LAP-B, LAP-D, PPP			
Физический	Коаксиал, STP, UTP, оптоволокно, радиоволны			

12

Стек OSI

Следует четко различать модель OSI и стек OSI. В то время как модель OSI является концептуальной схемой взаимодействия открытых систем, стек OSI представляет собой набор вполне конкретных спецификаций протоколов. В отличие от других стеков протоколов стек OSI полностью соответствует модели OSI, он включает спецификации протоколов для всех семи уровней взаимодействия, определенных в этой модели. На нижних уровнях стек OSI поддерживает Ethernet, Token Ring, FDDI, протоколы глобальных сетей, X.25 и ISDN, — то есть использует разработанные вне стека протоколы нижних уровней, как и все другие стеки. Протоколы сетевого, транспортного и сеансового уровней стека OSI специфицированы и реализованы различными производителями, но распространены пока мало. Наиболее популярными протоколами стека OSI являются прикладные протоколы. К ним относятся: протокол передачи файлов PTAM, протокол эмуляции терминала VTP, протоколы справочной службы X.500, электронной почты X.400 и ряд других.

Протоколы стека OSI отличает большая сложность и неоднозначность спецификаций. Эти свойства явились результатом общей политики разработчиков стека, стремившихся учесть в своих протоколах все случаи жизни и все существующие и появляющиеся технологии.

Из-за своей сложности протоколы OSI требуют больших затрат вычислительной мощности центрального процессора, что делает их наиболее подходящими для мощных машин, а не для сетей персональных компьютеров.

Стек OSI — международный, независимый от производителей стандарт. Его поддерживает правительство США в своей программе GOSIP, в соответствии с которой все компьютерные сети,

устанавливаемые в правительственных учреждениях США после 1990 года, должны или непосредственно поддерживать стек OSI, или обеспечивать средства для перехода на этот стек в будущем. Одним из крупнейших производителей, поддерживающих OSI, является компания AT&T, ее сеть Stargroup полностью базируется на этом стеке.

Стек TCP/IP был разработан по инициативе Министерства обороны США более 20 лет назад. Большой вклад в развитие стека TCP/IP, который получил свое название по популярным протоколам IP

и TCP, внес университет Беркли, реализовав протоколы стека в своей версии ОС UNIX. Популярность этой операционной системы привела к широкому распространению протоколов TCP, IP и других протоколов стека. Сегодня этот стек используется для связи компьютеров всемирной информационной сети Internet, а также в огромном числе корпоративных сетей.

Стек TCP/IP на нижнем уровне поддерживает все популярные стандарты физического и канального уровней: для локальных сетей — это Ethernet, Token Ring, FDDI, для глобальных — протоколы работы на аналоговых коммутируемых и выделенных линиях SLIP, PPP, протоколы территориальных сетей X.25 и ISDN.

Основными протоколами стека, давшими ему название, являются протоколы IP и TCP. Эти протоколы в терминологии модели OSI относятся к сетевому и транспортному уровням соответственно. IP обеспечивает продвижение пакета по составной сети, а TCP гарантирует надежность его доставки.

За долгие годы использования в сетях различных стран и организаций стек TCP/IP вобрал в себя большое количество протоколов прикладного уровня. К ним относятся такие популярные протоколы, как протокол пересылки файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы службы WWW и многие другие.

Сегодня стек TCP/IP представляет собой один из самых распространенных стеков транспортных протоколов вычислительных сетей. Только в сети Internet объединено около 10 миллионов компьютеров по всему миру, которые взаимодействуют друг с другом с помощью стека протоколов TCP/IP.

Стремительный рост популярности Internet привел и к изменениям в расстановке сил в мире коммуникационных протоколов — протоколы TCP/IP, на которых построен Internet, стали быстро теснить бесспорного лидера прошлых лет — стек IPX/SPX компании Novell. Сейчас любая промышленная операционная система обязательно включает программную реализацию этого стека в своем комплекте поставки.

Хотя протоколы TCP/IP неразрывно связаны с Internet и каждый из многомиллионной армады компьютеров Internet работает на основе этого стека, существует большое количество локальных, корпоративных и территориальных сетей, непосредственно не являющихся частями Internet, в которых также используют протоколы TCP/IP. Чтобы отличать их от Internet, эти сети называют сетями TCP/IP или просто IP-сетями.

Поскольку стек TCP/IP изначально создавался для глобальной сети Internet, он имеет много особенностей, дающих ему преимущество перед другими протоколами, когда речь заходит о построении сетей, включающих глобальные связи. В частности, очень полезным свойством, делающим возможным применение этого протокола в больших сетях, является его способность фрагментировать пакеты. Действительно, большая составная сеть часто состоит из сетей, построенных на совершенно разных принципах. В каждой из этих сетей может быть установлена собственная величина максимальной длины единицы передаваемых данных (кадра). В таком случае при переходе из одной сети, имеющей большую максимальную длину, в сеть с меньшей максимальной длиной может возникнуть необходимость деления передаваемого кадра на несколько частей. Протокол IP стека TCP/IP эффективно решает эту задачу.

Другой особенностью технологии TCP/IP является гибкая система адресации, позволяющая более просто по сравнению с другими протоколами аналогичного назначения включать в интерсеть сети других технологий. Это свойство также способствует применению стека TCP/IP для построения больших гетерогенных сетей.

В стеке TCP/IP очень экономно используются возможности широковещательных рассылок. Это свойство совершенно необходимо при работе на медленных каналах связи, характерных для территориальных сетей.

Однако, как и всегда, за получаемые преимущества надо платить, и платой здесь оказываются высокие требования к ресурсам и сложность администрирования IP-сетей. Мощные функциональные возможности протоколов стека TCP/IP требуют для своей реализации высоких вычислительных затрат. Гибкая система адресации и отказ от широковещательных рассылок приводят к наличию в IP-сети различных централизованных служб типа DNS, DHCP и т. п. Каждая из этих служб направлена на облегчение администрирования сети, в том числе и на облегчение конфигурирования оборудования, но в то же время сама требует пристального внимания со стороны администраторов.

Стек IPX/SPX

Этот стек является оригинальным стеком протоколов фирмы Novell, разработанным для сетевой операционной системы NetWare еще в начале 80-х годов. Протоколы сетевого и сеансового уровней Internetwork Packet Exchange (IPX) и Sequenced Packet Exchange (SPX), которые дали название стеку, являются прямой адаптацией протоколов XNS фирмы Xerox, распространенных в гораздо меньшей степени, чем стек IPX/SPX. Популярность стека IPX/SPX непосредственно связана с операционной системой Novell NetWare.

Многие особенности стека IPX/SPX обусловлены ориентацией ранних версий ОС NetWare на работу в локальных сетях небольших размеров, состоящих из персональных компьютеров со скромными ресурсами. Понятно, что для таких компьютеров компании Novell нужны были протоколы, на реализацию которых требовалось бы минимальное количество оперативной памяти (ограниченной в IBM-совместимых компьютерах под управлением MS-DOS объемом 640 Кбайт) и которые бы быстро работали на процессорах небольшой вычислительной мощности. В результате протоколы стека IPX/SPX до недавнего времени хорошо работали в локальных сетях и не очень — в больших корпоративных сетях, так как они слишком перегружали медленные глобальные связи широковещательными пакетами. Это обстоятельство, а также тот факт, что стек IPX/SPX является собственностью фирмы Novell и на его реализацию нужно получать лицензию (то есть открытые спецификации не поддерживались), долгое время ограничивали распространенность его только сетями NetWare. Однако с момента выпуска версии NetWare 4.0 Novell внесла и продолжает вносить в свои протоколы серьезные изменения, направленные на их адаптацию для работы в корпоративных сетях. Сейчас стек IPX/ SPX реализован не только в NetWare, но и в нескольких других популярных сетевых ОС, например SCO UNIX, Sun Solaris, Microsoft Windows NT.

Стек NetBIOS/SMB

Этот стек широко используется в продуктах компаний IBM и Microsoft. На физическом и канальном уровнях этого стека используются все наиболее распространенные протоколы Ethernet, Token Ring, FDDI и другие. На верхних уровнях работают протоколы NetBEUI и SMB.

Протокол NetBIOS (Network Basic Input/Output System) появился в 1984 году как сетевое расширение стандартных функций базовой системы ввода/вывода (BIOS) IBM PC для сетевой программы PC Network фирмы IBM. В дальнейшем этот протокол был заменен так называемым протоколом расширенного пользовательского интерфейса NetBEUI — NetBIOS Extended User Interface. С его помощью невозможна маршрутизация пакетов. Это ограничивает применение протокола NetBEUI локальными сетями, не разделенными на подсети, и делает невозможным его использование в составных сетях.

Протокол SMB (Server Message Block) выполняет функции сеансового, представительного и прикладного уровней. На основе SMB реализуется файловая служба, а также службы печати и передачи сообщений между приложениями.

На слайде показано соответствие некоторых, наиболее популярных протоколов уровням модели OSI. Часто это соответствие весьма условно, так как модель OSI — это только руководство к действию, причем достаточно общее, а конкретные протоколы разрабатывались для решения специфических задач, причем многие из них появились до разработки модели OSI. Чаще всего в стеке явно выделяются 3-4 уровня: уровень сетевых адаптеров, в котором реализуются протоколы физического и канального уровней, сетевой уровень, транспортный уровень и уровень служб, вбирающий в себя функции сеансового, представительного и прикладного уровней.

Различия локальных и глобальных сетей

- Протяженность, качество и способ прокладки линий связи
- Сложность методов передачи и оборудования
- Сложность обмена данными
- Разнообразие услуг
- Оперативность выполнения запросов
- Разделение каналов
- Использование метода коммутации пакетов
- Масштабируемость

13

Различия локальных и глобальных сетей.
В последнее время эти отличия становятся все менее заметные, тем не менее они существуют.

Протяженность, качество и способ прокладки линий связи. Класс локальных вычислительных сетей по определению отличается от класса глобальных сетей небольшим расстоянием между узлами сети. Это в принципе делает возможным использование в локальных сетях качественных линий связи: коаксиального кабеля, витой пары, оптоволоконного кабеля, которые не всегда доступны (из-за экономических ограничений) на больших расстояниях, свойственных глобальным сетям. В глобальных сетях часто применяются уже существующие линии связи (телеграфные или телефонные), а в локальных сетях они прокладываются заново.

Сложность методов передачи и оборудования. В условиях низкой надежности физических каналов в глобальных сетях требуются более сложные, чем в локальных сетях, методы передачи данных и соответствующее оборудование. Так, в глобальных сетях широко применяются модуляция, асинхронные методы, сложные методы контрольного суммирования, квитирование и повторные передачи искаженных кадров. С другой стороны, качественные линии связи в локальных сетях позволили упростить процедуры передачи данных за счет применения немодулированных сигналов и отказа от обязательного подтверждения получения пакета.

Скорость обмена данными. Одним из главных отличий локальных сетей от глобальных является наличие высокоскоростных каналов обмена данными между компьютерами, скорость которых (10, 100 Мбит/с) сравнима со скоростями работы устройств и узлов компьютера — дисков, внутренних шин обмена данными и т. п. За счет этого у пользователя локальной сети, подключенного к удаленному разделяемому ресурсу (например, диску сервера), складывается впечатление, что он пользуется этим диском, как «своим». Для глобальных сетей типичны гораздо более низкие скорости передачи данных — 9600, 28800, 33600 бит/с, 56 и 64 Кбит/с и только на магистральных каналах — до 2 Мбит/с.

Разнообразие услуг. Локальные сети предоставляют, как правило, широкий набор услуг — это различные виды услуг файловой службы, услуги печати, услуги службы передачи факсимильных сообщений, услуги баз данных, электронная почта и другие, в то время как глобальные сети в основном предоставляют почтовые услуги и иногда файловые услуги с ограниченными возможностями — передачу файлов из публичных архивов удаленных серверов без предварительного просмотра их содержания.

Оперативность выполнения запросов. Время прохождения пакета через локальную сеть обычно составляет несколько миллисекунд, время же его передачи через глобальную сеть может достигать нескольких секунд. Низкая скорость передачи данных в глобальных сетях затрудняет реализацию служб для режима on-line, который является обычным для локальных сетей.

Разделение каналов. В локальных сетях каналы связи используются, как правило, совместно сразу несколькими узлами сети, а в глобальных сетях — индивидуально.

Использование метода коммутации пакетов. Важной особенностью локальных сетей является неравномерное распределение нагрузки. Отношение пиковой нагрузки к средней может составлять 100:1 и даже выше. Такой трафик обычно называют пульсирующим. Из-за этой особенности трафика в локальных сетях для связи узлов применяется метод коммутации пакетов, который для пульсирующего трафика оказывается гораздо более эффективным, чем традиционный для глобальных сетей метод коммутации каналов. Эффективность метода коммутации пакетов состоит в том, что сеть в целом передает в единицу времени больше данных своих абонентов. В глобальных сетях метод коммутации пакетов также используется, но наряду с ним часто применяется и метод коммутации каналов, а также некоммутируемые каналы — как унаследованные технологии некомпьютерных сетей.

Масштабируемость. «Классические» локальные сети обладают плохой масштабируемостью из-за жесткости базовых топологий, определяющих способ подключения станций и длину линии. При использовании многих базовых топологий характеристики сети резко ухудшаются при достижении определенного предела по количеству узлов или протяженности линий связи. Глобальным же сетям присуща хорошая масштабируемость, так как они изначально разрабатывались в расчете на работу с произвольными топологиями.

Требования, предъявляемые к современным вычислительным сетям

- Производительность
- Надежность и безопасность
- Расширяемость и масштабируемость
- Прозрачность
- Поддержка разных видов трафика
- Управляемость
- Совместимость

14

Еще раз: главное требование, предъявляемое к сетям – это предоставлять пользователям потенциальную возможность доступа к разделяемым ресурсам всех компьютеров, объединенных в сеть.

Все остальные требования – связаны с качеством выполнения этой задачи. Хотя все эти требования весьма важны, часто понятие «качество обслуживания» трактуется более узко – в него включаются только две самые важные характеристики сети – производительность и надежность.

Производительность характеризуется следующими параметрами:

- Время реакции сети – интегральная характеристика. Именно она имеется в виду, когда пользователь говорит «Сегодня сеть работает медленно». В общем случае определяется как интервал времени между возникновением запроса к сетевой службе и получением ответа на этот запрос. Значение этого показателя зависит и от типа службы, к которой идет обращение и от текущего состояния элементов сети, загруженность коммутаторов, маршрутизаторов и т.д. Поэтому имеет смысл использовать средневзвешенную оценку загруженности, усредняя этот показатель по пользователям, серверам, времени суток и т.д.
- Пропускная способность – объем данных, переданный в единицу времени.
- Средняя пропускная способность. Отношение общего объема переданных данных к времени передачи (час, день, неделя).
- Мгновенная пропускная способность (10 мс –1 сек)
- Максимальная пропускная способность – наибольшая мгновенная за интервал измерения.
- Общая пропускная способность сети – количество информации, переданное между всеми узлами сети за время наблюдения.
- Задержка передачи – это параметр по смыслу близок ко времени реакции, но характеризует только сетевые этапы обработки данных, без задержек обработки компьютерами сети. Т.е., это задержка между временем поступления пакета на вход сетевого устройства и временем появления на выходе.

Пропускная способность и задержка – величины независимые. Сеть может иметь высокую ПС, но вносить большие задержки при передаче каждого пакета. Пример – спутниковая связь. Пропускная способность – 2 Mbit/s, а задержка передачи не менее 0,24 с. Что определяется длиной канала в 72000 км.

Надежность и безопасность сети характеризуется:

- Коэффициентом готовности – это доля времени, в течение которого система может быть использована. Готовность может быть улучшена путем введения избыточности в структуру системы. Ключевые элементы должны существовать в нескольких экземплярах. Чтобы систему можно было отнести к высоконадежным, она должна как минимум обладать высокой готовностью, но этого недостаточно. Необходимо обеспечить
- Сохранность данных и защиту их от искажений. Кроме того, должна поддерживаться согласованность данных, например при резервировании данных на нескольких файловых серверах (ресурсах) должна постоянно обеспечиваться их идентичность.
- Вероятность доставки пакета без искажений либо другие показатели, напр. Вероятность потери пакета
- Безопасность от несанкционированного доступа. Это другой аспект надежности.
- Отказоустойчивость. Способность системы скрыть от пользователя выход из строя отдельных элементов. В отказоустойчивой системе отказ одного из элементов приводит к некоторому снижению качества работы, а не к полному останову. Так при отказе одного из файловых серверов увеличивается только время доступа к базе данных из-за уменьшения степени распараллеливания запросов, но в целом система будет выполнять свои функции.

Расширяемость и масштабируемость.

- Расширяемость означает возможность добавления отдельных элементов сети (пользователей, компьютеров, приложений, служб), замены аппаратуры, наращивания сегментов сети. Однако расширение всегда ограничивается некоторыми пределами. Например, локальная сеть Ethernet, построенная на коаксиальном кабеле обладает хорошей расширяемостью, в смысле что позволяет легко

подключить новые станции. Однако такая сеть имеет ограничения – при числе компьютеров 30-40 штук она перестает корректно работать. Наличие такого ограничения – признак плохой масштабируемость системы при хорошей расширяемости.

- Масштабируемость говорит о том, что сеть позволяет наращивать количество узлов и протяженности связей без ухудшения производительности. Для обеспечения масштабируемости чаще всего приходится применять дополнительное коммуникационное оборудование.

Прозрачность сети достигается в том случае, когда сеть представляется пользователю не как множество отдельных компьютеров, связанной сложной системой оборудования, а как единая вычислительная машина. Известный лозунг компании Sun Microsystems: «Сеть – это компьютер».

Прозрачность может быть достигнута на двух различных уровнях — на уровне пользователя и на уровне программиста. На уровне пользователя прозрачность означает, что для работы с удаленными ресурсами он использует те же команды и привычные ему процедуры, что и для работы с локальными ресурсами. На программном уровне прозрачность заключается в том, что приложению для доступа к удаленным ресурсам требуются те же вызовы, что и для доступа к локальным ресурсам. Прозрачность на уровне пользователя достигается проще, так как все особенности процедур, связанные с распределенным характером системы, маскируются от пользователя программистом, который создает приложение. Прозрачность на уровне приложения требует сокрытия всех деталей распределенности средствами сетевой операционной системы.

Сеть должна скрывать все особенности операционных систем и различия в типах компьютеров. Пользователь компьютера Macintosh должен иметь возможность обращаться к ресурсам, поддерживаемым UNIX-системой, а пользователь UNIX должен иметь возможность разделять информацию с пользователями Windows 95. Подавляющее число пользователей ничего не хочет знать о внутренних форматах файлов или о синтаксисе команд UNIX.

В настоящее время нельзя сказать, что свойство прозрачности в полной мере присуще многим вычислительным сетям, это скорее цель, к которой стремятся разработчики современных сетей.

Поддержка разных видов трафика.

Главной особенностью трафика, образующегося при динамической передаче голоса или изображения, является наличие жестких требований к синхронности передаваемых сообщений. При запаздывании сообщений будут наблюдаться искажения. В то же время трафик компьютерных данных характеризуется крайне неравномерной интенсивностью поступления сообщений в сеть при отсутствии жестких требований к синхронности доставки этих сообщений. Например, доступ пользователя, работающего с текстом на удаленном диске, порождает случайный поток сообщений между удаленным и локальным компьютерами, зависящий от действий пользователя по редактированию текста, причем задержки при доставке в определенных (и достаточно широких с компьютерной точки зрения) пределах мало влияют на качество обслуживания пользователя сети. Все алгоритмы компьютерной связи, соответствующие протоколы и коммуникационное оборудование были рассчитаны именно на такой «пульсирующий» характер трафика, поэтому необходимость передавать мультимедийный трафик требует внесения принципиальных изменений как в протоколы, так и оборудование.

Особую сложность представляет совмещение в одной сети традиционного компьютерного и мультимедийного трафика. Передача исключительно мультимедийного трафика компьютерной сетью хотя и связана с определенными сложностями, но вызывает меньшие трудности. А вот случай сосуществования двух типов трафика с противоположными требованиями к качеству обслуживания является намного более сложной задачей. Обычно протоколы и оборудование компьютерных сетей относят мультимедийный трафик к факультативному, поэтому качество его обслуживания оставляет желать лучшего. Сегодня затрачиваются большие усилия по созданию сетей, которые не ущемляют интересы одного из типов трафика. Наиболее близки к этой цели сети на основе технологии ATM, разработчики которой изначально учитывали случай сосуществования разных типов трафика в одной сети.

Управляемость.

Управляемость сети подразумевает возможность централизованно контролировать состояние основных элементов сети, выявлять и разрешать проблемы, возникающие при работе сети, выполнять анализ производительности и планировать развитие сети.

Хорошая система управления наблюдает за сетью и, обнаружив проблему, активизирует определенное действие, исправляет ситуацию и уведомляет администратора о том, что произошло и какие шаги предприняты. Одновременно с этим система управления должна накапливать данные, на основании которых можно планировать развитие сети. Наконец, система управления должна быть независима от производителя и обладать удобным интерфейсом, позволяющим выполнять все действия с одной консоли.

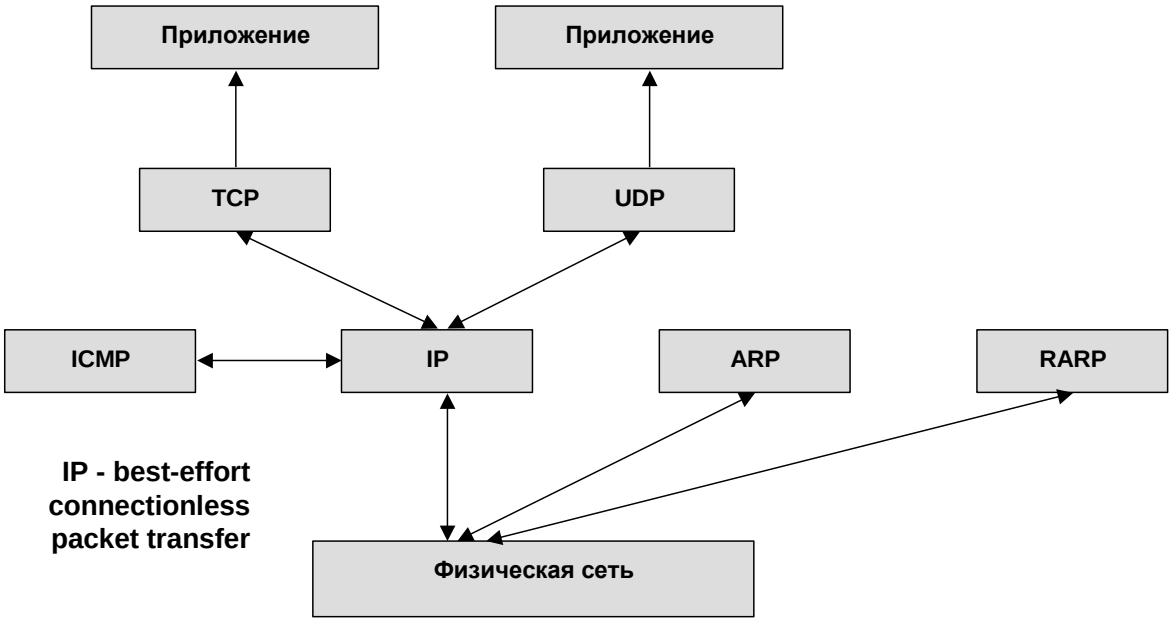
Полезность системы управления особенно ярко проявляется в больших сетях: корпоративных или публичных глобальных. Большинство существующих средств вовсе не управляют сетью, а всего лишь осуществляют наблюдение за ее работой. Они следят за сетью, но не выполняют активных действий, если с сетью что-то произошло или может произойти.

Совместимость.

Совместимость или интегрируемость означает, что сеть способна включать в себя самое разнообразное программное и аппаратное обеспечение, то есть в ней могут сосуществовать различные операционные системы, поддерживающие разные стеки коммуникационных протоколов, и работать аппаратные средства и приложения от разных производителей. Сеть, состоящая из разнотипных элементов, называется неоднородной или гетерогенной, а если гетерогенная сеть работает без проблем, то она является интегрированной. Основной путь построения интегрированных сетей — использование модулей, выполненных в соответствии с открытыми стандартами и спецификациями.

Лекция 3. Сети TCP/IP.

Архитектура сетей TCP/IP



Состав стека протоколов TCP/IP

Архитектура сетей TCP/IP.

В состав стека протоколов TCP/IP, кроме давших название этим сетям протоколов Transmission Control Protocol и Internet Protocol, входят: User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol) и ряд протоколов прикладного уровня, в частности TELNET, FTP, SMTP, SNMP и HTTP.

В отличие от 7-ми уровневой модели OSI протокольный стек TCP/IP разбит на 4 уровня. Сетевой уровень (IP) обеспечивает передачу информации через произвольную комбинацию сетей, использующих этот же набор протоколов. **IP-протокол предоставляет лишь один вид сервиса – передачу пакетов без предварительного установления соединения и настолько хорошо, насколько получиться** (best-effort connectionless packet transfer). Пакеты пересылаются между узлами коммутации без предварительного установления соединения; они маршрутизируются независимо, и пакеты одного приложения могут доставляться по разным маршрутам. Узлы коммутации, соединяющие смежные сети, могут испытывать перегрузки и уничтожать пакеты. Ответственность за восстановление утерянных пакетов и надлежащий порядок их передачи приложению лежит на транспортном уровне, который представлен протоколами TCP и UDP.

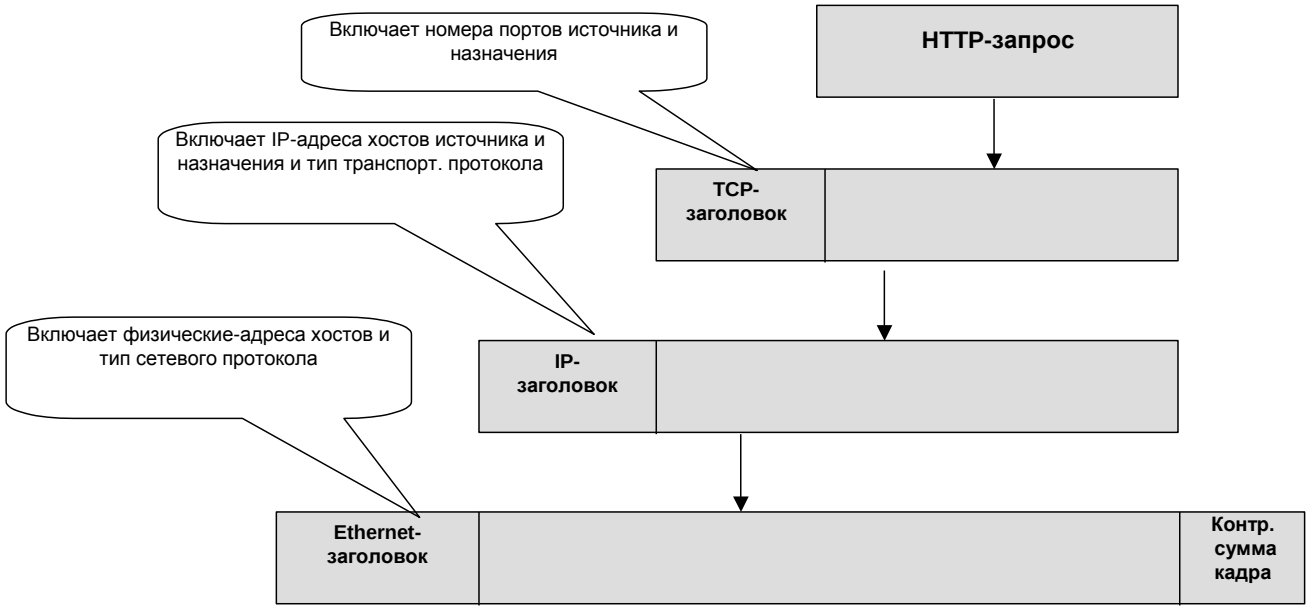
Разнообразие требований сетевых приложений обусловило необходимость двух протоколов транспортного уровня. Так, например, приложения передачи файлов и web (FTP, HTTP) для пересылки своих сообщений используют TCP, в то время как приложения управления сетевыми устройствами, служба имен (SNMP, DNS), потоковые приложения реального времени используют в качестве транспортного протокола UDP. Далее будем называть протокольные блоки TCP сегментами, а блоки UDP – дейтограммами.

Сетевой уровень (протокол IP) мультиплексирует протокольные блоки транспортного уровня в IP-потоки; при этом сегменты транспортного уровня могут фрагментироваться (если они превышают максимально допустимый размер, определяемый канальным протоколом). Протокольные блоки IP обычно называют пакетами.

После вычисления маршрута передачи пакета, посредством протокола ARP определяется физический адрес следующего на маршруте хоста и пакет направляется на физический уровень сети. Иногда бывает необходимо решить обратную задачу, то есть по заданному физическому адресу определить логический сетевой адрес устройства. В частности, эта проблема актуальна для процедуры загрузки бездисковых станций, когда такая станция рассылает в широковещательном режиме запрос, содержащий ее физический адрес и «просьбу» сообщить ей логический сетевой адрес. Этот запрос обрабатывается сервером RARP, который и передает пославшей его станции требуемую информацию.

Физический уровень TCP/IP сети может использовать любую технологию канального уровня – Ethernet, Token Ring, ATM, PPP и т.д. Но для обеспечения прозрачности физического уровня необходимо, чтобы на сетевом уровне были предусмотрены процедуры дефрагментации пакетов до размера, разрешенного соответствующим протоколом канального уровня. Обратная процедура, т.е. объединение пакетов малых размеров до величины, приемлемой на канальном уровне, не предусматривается.

Архитектура сетей TCP/IP



Инкапсуляция протокольных блоков в TCP/IP стеке

3

Протокольные блоки вышележащих уровней инкапсулируются в протокольные блоки нижележащих уровней так, как это показано на слайде. При этом, блок каждого уровня содержит специфическую информацию, позволяющую точно адресовать его. Так, сегмент TCP (UDP-дейтограмма) содержит в своем заголовке номер **порта**, однозначно определяющий приложение, которому он принадлежит. IP-пакет адресуется **логическим адресом** хоста, также однозначно определяющим его в распределенной сети. Кадр канального уровня включает в себя **физический адрес** ближайшего на маршруте доставки хоста, которому необходимо передать пакет.

IP-протокол.

Основной протокол стека TCP/IP, - Internet Protocol (IP), - по своим функциям соответствует сетевому уровню модели взаимодействия открытых систем (OSI). Механизмы протокола, описанные в документе RFC 791, обеспечивают ненадежную доставку пакетов данных между сетевыми устройствами (устройствами, имеющими сетевой адрес) в режиме без предварительного установления соединения (дейтограммный сервис). Этот тип сервиса часто называют сервисом «настолько хорошо, как получиться» (best effort service), что отражает отсутствие в протоколе процедур контроля доставки пакетов. Решение задачи надежности доставки возлагается на протоколы верхних уровней, главным образом на TCP. Основными функциями протокола IP являются:

- формирование пакетов из сегментов транспортного уровня, с предварительной фрагментацией (если необходимо) последних;
- обеспечение логической адресации сетевых устройств
- поддержка процесса маршрутизации
- продвижение пакетов от одного узла коммутации до другого.

Структура IP – пакета

Формат заголовка IP-пакета

4

Структура IP-пакета.

Функции IP-протокола реализуются посредством специальной структуры заголовка пакета, формат которого приведен на слайде.

Заголовок содержит поля фиксированной длины (первые 20 байт) и поле переменной длины (поле опций), размер которого может достигать 40 байт. Для выравнивания этого поля по 32 битной границе

предусмотрено поле «Выравнивание» (Padding) которое заполняется нулями. Рассмотрим назначение полей заголовка.

Версия (Version) – поле определяет номер версии протокола. В настоящее время используется версия 4 и ведется активная подготовка к переходу на версию 6. Версия 5 описывает протокол ST2, разработанный для передачи данных потоковых приложений реального времени. Поле проверяется перед обработкой пакета и пакеты несогласующейся с протокольным стеком приемника версией, отбрасываются. Одновременно, включение версии в каждую дейтограмму позволяет использовать разные, но согласующиеся, версии на разных хостах.

Длина заголовка (Internet Header Length, IHL) - Поле определяет длину заголовка, измеренную в 32-битных словах. Корректный заголовок имеет длину не менее 5 слов. Длина поля опций (в 32-разрядных словах) может быть определена как значение этого поля минус 5.

Тип сервиса (Type of service, ToS) – определяет тип требуемого обслуживания пакета. Первые три бита задают уровень приоритета обслуживания (0-7), 3-5 биты определяют требования к задержке (какая получится, низкая), уровень пропускной способности (обычный, высокий) и надежность доставки (какая получится, высокая). Практически, большая часть маршрутизаторов игнорирует данные этого поля. Однако в настоящее время в связи с разработкой механизмов обеспечения в IP-сетях служб с гаранированным качеством обслуживания делаются попытки использования значений этого поля.

Общая длина (Total length) - поле содержит общую длину пакета, размер которого не может превышать 65535 байт. Практически пакеты такой длины никогда не используются, поскольку технологии канального уровня накладывают свои ограничения. Так, Ethernet не допускает кадров с длиной более 1500 байт, FDDI – 4096 байт и т.д. В этой связи, протокол IP выполняет фрагментацию сегментов данных, поступающих к нему от TCP и UDP протоколов. Следует отметить, что маршрутизатор не выполняет сборку пакетов, даже если следующая сеть имеет параметр MTU (Maximum Transmission Unit), допускающий более крупные пакеты. Сборка пакетов в исходный сегмент производится на месте назначения.

Поля «**Идентификатор**», **«Флаги**» и «**Смещение фрагмента**» управляют процессом сборки сегмента.

Время жизни (Time to live, TTL) - поле, определяющее максимальное время, которое пакет может существовать в сети. Значение этого поля (в секундах) устанавливается при отправке пакета и уменьшается на единицу по мере прохождения им маршрутизаторов. При достижении нулевого значения этого поля пакет уничтожается. Максимальное значение поля – 255 секунд. Этот механизм помогает избежать перегрузок сети при возникновении ошибок в таблицах маршруизации, приводящих к образованию петель.

Структура IP – пакета

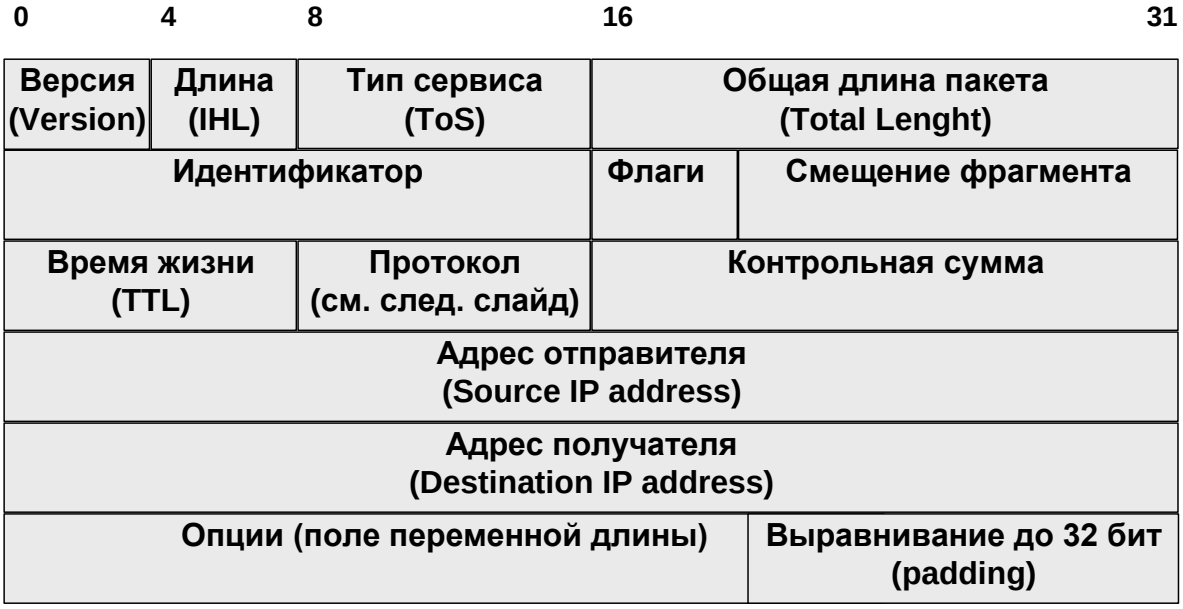
Значение поля	Ключевое слово	Протокол
0	**Reserved**	**Зарезервировано**
1	**ICMP**	**Internet Control Message Protocol**
2	**IGMP**	**Internet Group Management Protocol**
4	**IP**	**Internet Protocol**
6	**TCP**	**Transmission Control Protocol**
17	**UDP**	**User Datagram Protocol**
89	**OSPF**	**Open Shortest Path First**

Поле «Протокол» заголовка IP-пакета

5

Протокол (Protocol) – поле указывает модулю какого протокола (TCP, UDP, ICMP) передать полученный IP-пакет. На слайде 5. приведены значения этого поля для некоторых из протоколов. В дальнейшем нас будут интересовать TCP, UDP, ICMP.

Структура IP – пакета



Формат заголовка IP-пакета

4

Контрольная сумма (Header checksum) – поле содержит значение контрольной суммы**,** рассчитанной только по заголовку. Поскольку значения некоторых полей заголовка изменяются по мере прохождения пакета по маршруту (поле TTL, например), то значения рассматриваемого поля проверяются и пересчитываются на каждом маршрутизаторе. Этот механизм является единственным средством обеспечения достоверности передачи, содержащимся в протоколе IP.

Адрес отправителя (Source IP address) и **Адрес получателя (Destination IP address)** – поля одинаковой длины (32 бита), содержащие соответствующие адреса. Правила адресации в IP-сетях будут рассмотрены далее.

Опции (Options) – необязательное поле, используемое при отладке сетей и для запроса определенных специфических процедур обработки. В настоящее время используется крайне редко. В связи с разработкой новых протоколов, обеспечивающих большую гибкость в обработке IP-трафика, возможность использования этих полей вновь стала предметом обсуждения комитетов по стандартизации.

При получении пакета маршрутизатор вычисляет контрольную сумму заголовка пакета и, если она не совпадает со значением поля «Контрольная сумма», то пакет отбрасывается. При положительном результате проверки, производится изменение некоторых полей и рассчитывается новое значение поля «Контрольная сумма». Затем по таблице маршрутизации определяется адрес следующего маршрутизатора, на который должен быть направлен этот пакет, и он передается на соответствующий интерфейс.

Адресация в сетях IP

Классификация IP адресов

Класс сетей	Значение первого байта адреса	Количество сетей	Количество хостов в сети класса	Удельный вес класса в адресном пространстве, %
A	001 – 126	126	16 777 214	50
B	128 – 191	16 384	65 534	25
C	192 - 223	2 097 152	254	12,5

6

Адресация в сетях IP.

Для идентификации каждого компьютера в IP-сети необходима система их адресации. При этом учитывается, что сетевые устройства (компьютер, маршрутизатор и т.д.) могут иметь несколько сетевых интерфейсов, и каждый из них должен иметь уникальный адрес. Если обратиться к аналогии обычной адресной системы (улица, дом, квартира), то становиться ясной целесообразность построения системы сетевой адресации по иерархии «сеть-интерфейс». Принятая в сетях IP система адресации описана в документах RFC 990 и RFC 997. Каждое сетевое устройство имеет адреса трех типов:

1. Физический адрес узла, определяемый используемой технологией канального уровня. Для Ethernet – это MAC-адрес его сетевой карты, назначаемый фирмой-производителем. Он представляет собой шести-байтовое число, первые три байта которого однозначно определяют фирму-производителя, а последние три байта – уникальны для каждой карточки, произведенной в рамках данной фирмы.
2. IP-адрес, состоящий из 4-х байтов, и также являющийся совершенно уникальным.
3. Символьный идентификатор – имя, назначаемое по определенным правилам и являющееся полным эквивалентом IP-адреса.

IP-адрес строится по двухуровневой иерархии, т.е. он объединяет в себе адрес сети и адрес хоста. Разделение сетевого адреса на 2 части имеет большой практический смысл, ибо позволяет магистральным маршрутизаторам существенно сократить размер своих таблиц коммутации, формируя их на основании только сетевой части адреса назначения. Для удовлетворения потребностей адресации сетей различного масштаба были введены несколько классов сетей, отличающиеся размером полей, отводимых для указания номера сети и номера хоста. При этом, размер поля полного адреса всегда равен 32 битам. Структура адресов сетей разных классов приведена на слайде.

IP-адрес обычно записывается в форме 4-х трехразрядных десятичных чисел, разделенных точкой. Каждое из этих десятичных чисел соответствует одному байту двоичного представления адреса. Так, например, адрес 10000000 10000111 01000100 00000101 в десятичном представлении имеет вид 128.135.68.5. В этом случае, т.к. первые два бита адреса – 10, то это адрес хоста, принадлежащего сети класса **B** и, следовательно, левые 16 бит являются адресом сети, а правые16 бит – адресом хоста.

Некоторые адреса являются зарезервированными и не могут присваиваться хостам. Так, адрес 127.x.x.x (x – означает любое число, обычно 0) зарезервирован для обратной связи, используемой при тестировании взаимодействия процессов на одной сетевой станции. Когда приложение использует этот адрес в качестве адреса назначения, стек TCP/IP данного хоста возвращает данные приложению, ничего не передавая на физический интерфейс. Поэтому адреса, начинающиеся на 127, запрещается присваивать сетевым устройствам. Другим зарезервированным адресом является, так называемый, широковещательный адрес, содержащий 1, или 0, во всех своих битах. Пакет с адресом назначения 255.255.255.255 (1.1.1.1) будет доставлен всем устройствам сети, к которой принадлежит узел-отправитель, но маршрутизаторы такие пакеты не обрабатывают. Существует и направленное широковещание – способ адресации, при котором один пакет, отосланный в определенную сеть, будет доставлен всем ее хостам. Такой пакет должен содержать корректный адрес сети и иметь все биты адреса хоста равными 1. Так, например, пакет с адресом 184.90.255.255 будет доставлен всем станциям сети класса **B**, имеющей адрес 184.90.

Из рисунка вверху хорошо видно, что значения первых четырех битов адреса однозначно определяют обе границы его сетевой части. Нетрудно сосчитать, что максимальное число сетей класса **A** равно $2^6+2^5+ \ldots.+2^0 -1 =126$ (сетевой префикс, состоящий из одних нулей недопустим). Каждая сеть класса **A** может содержать $2^{24}-2 = 16\ 777\ 214$ устройств. В целом, адресный блок сетей класса **A** занимает около 50% общего адресного пространства. В таблице внизу слайда содержатся аналогичные показатели для сетей класса **B** и **C.**

Разбиение IP-сети на подсети.

Рассмотренная выше двухуровневая схема адресации оказалась недостаточно гибкой и в 1985 году была определена трехуровневая схема адресации (RFC 950). Необходимость перехода к такой системе адресации можно проиллюстрировать следующим примером. Организация получила сеть класса **B**, позволяющую адресовать 65000 хостов. Пока в сети работали относительно немного станций – она функционировала благополучно. Но с ростом числа активных хостов неизбежно рос и широковещательный трафик, поскольку этот тип пакетов активно используется в служебных целях многими протоколами. Это обстоятельство неизбежно вело к снижению эффективной производительности сети. Кроме того, управление несколькими десятками тысяч подключений из единого административного центра представляет собой очень трудную задачу. К этим проблем следует добавить и то, что любая организация развивается в направлении роста самостоятельности своих подразделений, что также неизбежно должно отражаться и в структуре сети. Возможность же структурирования сети организации за счет получения дополнительных номеров сетей была быстро исчерпана и резкая нехватка адресного пространства стала реальностью еще во второй половине 80-х годов. Кроме того, рост числа используемых сетей вел к росту таблиц маршрутизации магистральных (межсетевых) маршрутизаторов и, следовательно, к снижению их производительности.

Перечисленные проблемы в значительной степени связаны с проблемой масштабируемости сети, и они, в значительной степени, были разрешены посредством введения в иерархию адресации третьего уровня. Этот уровень, получивший название «уровень подсети», был выделен в пространстве адресов хоста

Разбиение IP-сети на подсети

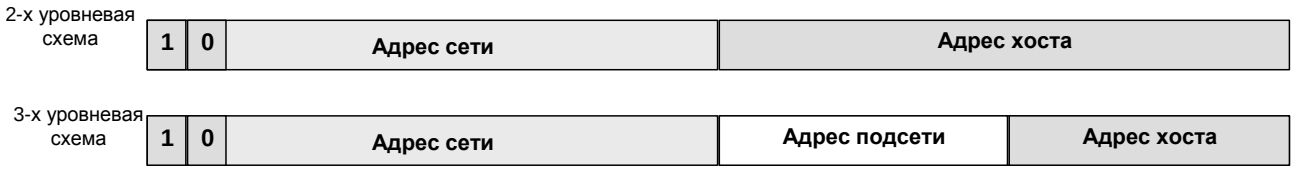


Схема адресации с введением подсетей

	Подсеть	Маска	Эквивалентный адрес подсети	Число хостов в подсети
Исходная сеть	**193.10.1.0**	**255.255.255.0**	**193.10.1.0/24**	**254**
Подсеть 1	**193.10.1.0**	**255.255.255.128**	**193.10.1.0/25**	**126**
Подсеть 2	**193.10.1.128**	**255.255.255.192**	**193.10.1.128/26**	**62**
Подсеть 3	**193.10.1.192**	**255.255.255.224**	**193.10.1.192/27**	**30**
Подсеть 4	**193.10.1.224**	**255.255.255.224**	**193.10.1.224/27**	**30**

Разбиение сети класса «С» на подсети

При этом поля адреса сети и адреса подсети в совокупности называют расширенным сетевым префиксом. В рассмотренном выше примере, выделение в пространстве адреса хоста 6 разрядов для адреса подсети позволяет организовать 62 подсети, содержащих до 1022 хостов каждая. Такое разбиение на подсети не требует согласования со специальным служебным органом Интернет, регулирующим распределение адресного пространства, Network Information Center.

Отрицательным следствием введения подсетей стало усложнение процедуры определения адреса хоста. Действительно, сетевые устройства используют старшие биты сетевого адреса для определения класса сети, после чего, в случае двухуровневой иерархии, легко находится граница между битами сетевого адреса и адреса хоста. В трехуровневой системе адресации этот прием работать не сможет. Для определения адреса подсети и адреса хоста потребовалось ввести **сетевую маску –** 32-битный адрес, в котором все биты, соответствующие битам расширенного сетевого префикса, установлены в 1, а соответствующие битам адреса хоста – в 0. Так, например, пусть необходимо в сети класса **B** 132.10 выделить подсеть, содержащую не более 100 хостов. Для адресации такого числа сетевых устройств достаточно располагать 7 битами (2^7 =128), которые и отводятся для адреса хоста; оставшиеся 9 бит отводятся для номера подсети (их можно организовать 2^9 -2=510) а старшие 16 бит – это адрес сети.

Таким образом, маска подсети, в которой находятся хосты с адресами:

132.10.12.129, 132.10.12.130,….,132.10.12.254

будет иметь вид

11111111 11111111 11111111 10000000 (255.255.255.128).

Если маршрутизатор получит пакет с адресом назначения

10000100 00001010 00001100 10110000 (132.10.12.176)

и выполнит по отношению к нему и сетевой маске операцию логического И, то результат будет содержать номер подсети. В данном случае получаем:

10000100 00001010 00001100 10000000,

что соответствует адресу подсети 132.10.12.128. Далее, по таблице маршрутизации будет найден следующий узел на пути к этой подсети, и пакет отправится на соответствующий интерфейс.

Необходимо заметить, что информация о маске подсети IP-пакетом не переносится (хост-источник о структуре сети, в которой находится получатель, ничего не знает). Передача этой информации является задачей протоколов маршрутизации, или она задается статически при конфигурировании маршрутизатора.

В стандартах, описывающих современные протоколы маршрутизации, часто делается ссылка на длину расширенного префикса сети, а не на маску подсети. В такой записи адрес устройства в рассмотренном выше примере имеет вид 132.10.12.176/25. В качестве примера нумерации подсетей в таблице приведено разбиение сети класса **C** на подсети. Указанное в таблице число хостов в каждой из подсетей на два меньше возможного числа адресов, поскольку адреса, в которых все биты поля адреса хоста установлены в единицу и в ноль, являются зарезервированными и используются как широковещательные для данной подсети. Так при подсетях, приведенных в таблице., адрес 193.10.1.0 является широковещательным лишь для подсети 193.10.1.0/24, а не для всех подсетей. Аналогично, адрес 193.10.1.255 является широковещательным только для подсети 193.10.1.224/27.

Введение подсетей, решив проблемы масштабирования адресного пространства, потребовало определенного усложнения протоколов маршрутизации, которые должны обрабатывать (и переносить) не только адрес сетевого устройства, но и его маску. В настоящее время все широко используемые протоколы маршрутизации (RIP-2, IS-IS, OSPF) переносят эту информацию.

IP маршрутизация.

Каждый хост (станция, маршрутизатор) ведет свои маршрутные таблицы, которые и определяют порядок обработки IP-пакетов. Передача пакетов между конечными станциями, требует взаимодействия

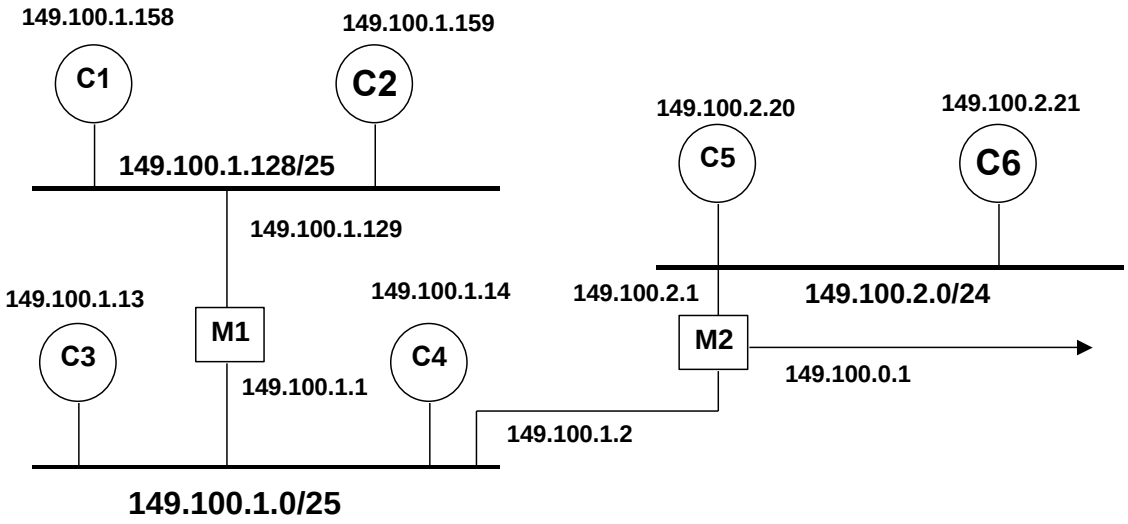
IP-модулей программного обеспечения этих станций и маршрутизаторов, связывающих сети, в которых они (станции) находятся. Рассмотрим, как обрабатывается пакет на этих сетевых устройствах.

Если в таблице маршрутизации станции-отправителя указано, что станция назначения является непосредственно присоединенной к той же ЛВС, то из таблицы физических адресов, которая ведется на каждой сетевой станции, извлекается физический адрес узла назначения, пакет инкансулируется в кадр канального протокола и передается к станции назначения. Если таблица маршрутизации станции-отправителя не содержит искомый сетевой адрес, то пакет отправляется по адресу маршрутизатора, который был указан при конфигурировании станции в качестве шлюза по умолчанию (default router, default gateway). Этот шлюз обязательно имеет физический интерфейс в той же ЛВС, что и станция-отправитель. При получении пакета маршрутизатор проверяет, не совпадает ли адрес назначения этого пакета с его собственным IP-адресом. Если это так, то пакет передается модулю протокола, указанного в поле «Протокол» заголовка пакета. В противном случае, маршрутизатор посредством своей таблицы определяет адрес следующего хоста, которому он должен передать этот пакет, и свой интерфейс, на который следует его направить.

Каждая строка в таблице маршрутизации содержит следующую информацию: IP-адрес сети (узла) назначения, IP-адрес следующего маршрутизатора, способного обеспечить передачу пакета в эту сеть (этому узлу), имя выходного интерфейса и некоторые флаги. Флаги содержат уточняющую информацию о каждой записи в таблице. Так например, флаг **H** определяет, является ли данная строка таблицы маршрутом к хосту (**H**=1), или к сети (**H**=0); флаг **G** уточняет, является ли она маршрутом к другому маршрутизатору (**G**=1), или определяет путь к непосредственно подключенной станции (**G**=0).

Поиск в таблице маршрутизации ведется следующим образом. Прежде всего, сканируется ее первый столбец с целью нахождения записи, точно соответствующей адресу назначения пакета. Если такая обнаруживается, то пакет отправляется по адресу, указанному в столбце «Следующий маршрутизатор». В противном случае, ведется поиск строки, содержащей адрес сети назначения. Если такой записи нет, то ищется строка, определяющая маршрут по умолчанию, т.е. маршрут к узлу с более полной информацией о траектории передачи этого пакета. Если не один из перечисленных вариантов не реализуем, то пакет уничтожается, а узлу-отправителю отправляется сообщение «Хост недостижим». Рассмотрим таблицы маршрутизации в сети, изображенной на след. слайде.

IP – маршрутизация



Пример сети, объединенной маршрутизаторами

Таблица маршрутизации **C6** :

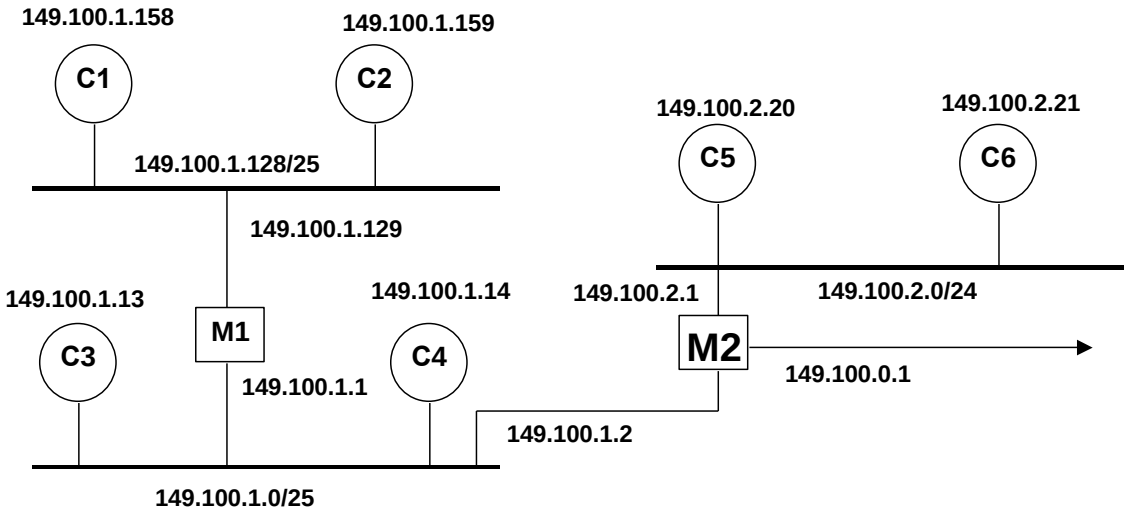
Адрес назначения	Маска сети	Следующий узел	Флаг	Интерфейс
127.0.0.0	255.0.0.0	127.0.0.1	H	lo0
Default	0.0.0.0	149.100.2.1	G	Eth0
149.100.2.0	255.255.255.0	149.100.2.21		Eth0

8

Эта сеть содержит три подсети: 149.100.1.0/25 149.100.1.128/25, 149.100.2.0/24, которые объединены двумя маршрутизаторами M1 и M2. При этом маршрутизатор M2 является шлюзом в Интернет. Адреса интерфейсов маршрутизаторов показаны на рисунке. Предположим, что станция C6 имеет пакет с адресом назначения 149.100.1.159. Таблица маршрутизации C6 выглядит следующим образом (таблица внизу слайда):

Первая строка таблицы определяет закольцовывающий интерфейс. Вторая строка описывает маршрут по умолчанию и флаг G уточняет, что узел 149.100.2.1 является маршрутизатором. Третья запись таблицы не имеет флага H, следовательно, она определяет сеть; нет в ней и флага G и это говорит о том, что определяется маршрут к непосредственно подключенной сети и интерфейсом к ней является внешний интерфейс станции C6, имеющий адрес 149.100.2.21. Прежде всего C6 производит поиск адреса станции (или сети) назначения в своей таблице. Поскольку ни тот, ни другой в ней не присутствуют, то пакет будет отправлен в соответствии с маршрутом по умолчанию на маршрутизатор M2 с адресом 149.100.2.1 через интерфейс Eth0.

IP – маршрутизация



Пример сети, объединенной маршрутизаторами

Таблица маршрутизации **M2** :

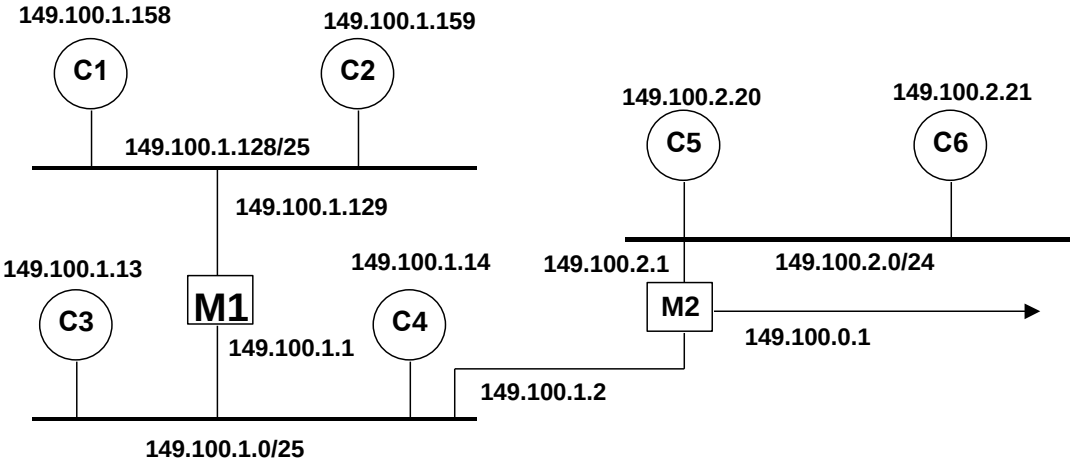
Адрес назначения	Маска	Следующий узел	Флаг	Интерфейс
127.0.0.0	255.0.0.0	127.0.0.1	H	lo0
default		149.100.0.1	G	Serial 01
149.100.2.0	255.255.255.0	149.100.2.1		Eth0
149.100.1.0	255.255.255.192	149.100.1.2		Eth1
149.100.1.128	255.255.255.192	149.100.1.1	G	Eth1

9

Таблица маршрутизации M2 может иметь следующий вид (внизу слайда).

Выполнив поиск в этой таблице маршрутизатор M2 найдет, что наиболее полно согласуется с адресом назначения маршрут, определенный в строке 5 и отправит пакет к маршрутизатору M1 через интерфейс Eth1.

IP – маршрутизация



Пример сети, объединенной маршрутизаторами

Таблица маршрутизации **M1** :

Адрес назначения	Маска	Следующий узел	Флаг	Интерфейс
127.0.0.0	255.0.0.0	127.0.0.1	H	lo0
default		149.100.1.2	G	Eth1
149.100.2.0	255.255.255.0	149.100.1.2	G	Eth1
149.100.1.0	255.255.255.192	149.100.1.1		Eth1
149.100.1.159	255.255.255.255	149.100.1.159		Eth0
149.100.1.128	255.255.255.192	149.100.1.129		Eth0

10

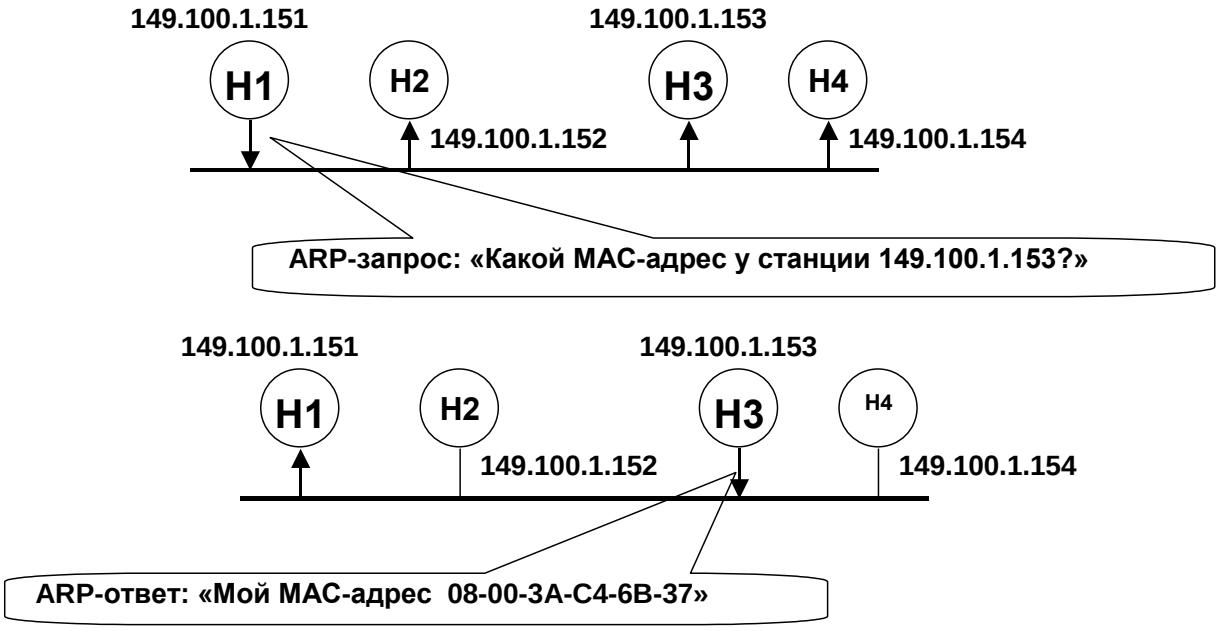
Таблица маршрутизации M1 представлена внизу в таблице.

В ней представлен маршрут с адресом назначения точно соответствующим адресу назначения пакета. Поэтому последний передается на интерфейс Eth0 и доставляется станции C2 с IP-адресом 149.100.1.159.

Разрешение IP-адресов в физические адреса сетевых устройств. Протокол ARP.

Для доставки IP-пакета к станции назначения, или от одного маршрутизатора к другому, необходимо передать его протоколу канального уровня, например Ethernet. Последний же «умеет» передавать кадры только по физическим адресам устройств, подключенных к среде передачи. В IP-сетях задачу преобразования сетевых адресов в физические решает протокол ARP (Address Resolution Protocol). Идея его функционирования иллюстрируется следующий слайд.

ARP (Address Resolution Protocol)



Разрешение IP-адреса в локальной сети

11

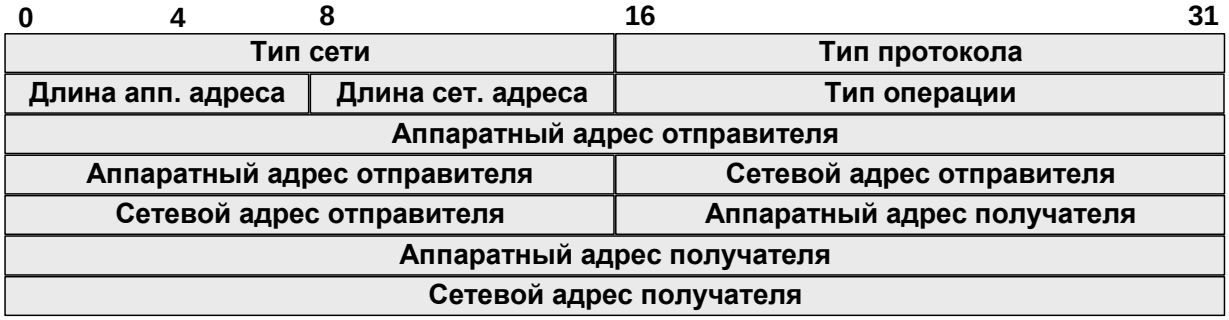
Пусть хост H1 хочет отослать пакет хосту H3, MAC-адрес которого не известен. Хост H1 генерирует так называемый ARP-запрос – специальный пакет, имеющий широковещательный адрес назначения. В теле этого запроса находится IP-адрес хоста, MAC-адрес которого необходимо узнать. Каждый хост сети получив такой пакет, сравнивает находящийся в нем IP-адрес со своим. Если совпадение обнаружено, то этот хост посылает запрашивающей станции ответный пакет (ARP-ответ), содержащий его физический адрес. На всех остальных станциях сети пакеты ARP-запросов уничтожаются. Для того, чтобы уменьшить количество ARP-запросов, каждое сетевое устройство имеет специальную буферную память, в которой хранится ARP-таблица. Последняя пополняется каждый раз, когда хост получает ARP-ответ.

В ARP-таблице могут быть как статические, так и динамические записи. Статические записи добавляются администратором и сохраняются в таблице до перезагрузки устройства. Кроме того, в таблице всегда содержится широковещательный адрес (%FFFFFFFFFFFF), который позволяет принимать широковещательные запросы. Динамические записи добавляются и удаляются автоматически. Каждая такая запись имеет потенциальное время жизни. После добавления записи в таблицу включается специальный таймер и, если в течении первых двух минут запись не используется, то она удаляется; в противном случае, время жизни такой записи составляет некое предустановленную величину (обычно 10 минут). Далее приведен фрагмент ARP-таблицы маршрутизатора.

ARP (Address Resolution Protocol)

IP-address	Port	Type	Media Address	Address Source
20.0.0.0	2	Broadcast	%FFFFFFFFFFFF	Static
10.0.0.0	1	Broadcast	%FFFFFFFFFFFF	Static
10.0.0.10	1	Local	%080002145DE6	Static
20.0.0.10	2	Local	%080002145DE5	Static
10.0.0.1	1	External	%080002A56BC0	ARP
20.0.0.1	2	External	%080002A719DD	ARP

Фрагмент ARP-таблицы маршрутизатора



Формат ARP-пакета

12

Протокол ARP достаточно универсален, и его можно применять в сетях, использующих любые технологии на сетевом и канальном уровнях. Заголовок ARP-пакета имеет формат, отличный от IP-заголовка и поэтому не передается маршрутизаторами. На рисунке внизу представлен формат сообщения ARP.

В поле «Тип сети» для Ethernet указывается значение 1. Для других типов сетей его значения определяются соответствующим стандартом. Поля «Тип протокола», «Длина аппаратного адреса» и «Длина сетевого адреса» обеспечивают отмеченную выше универсальность формата ARP-пакета. В поле «Тип операции» для ARP-запроса указывается 1, для ARP-ответа – 2.

Устройство, отправляющее ARP-запрос, заполняет в этом пакете все поля, кроме искомого аппаратного адреса. Значение этого поля заполняется станцией, опознавшей свой IP-адрес, указанный в этом пакете в поле «IP-адрес получателя».

ICMP – протокол обмена управляющей информацией

Тип	Содержание	Тип	Содержание
0	Ответ на эхо	11	Превышено время жизни пакета
3	Получатель недостижим	12	Ошибка параметров в пакете
4	Подавление источника	17	Запрос маски адреса
5	Изменение маршрута	18	Ответ на запрос маски адреса
8	Запрос эха		

Формат сообщений «Запрос эха» и «Ответ на эхо»

Выше упоминалось, что если маршрутизатор не может по каким-то причинам отправить пакет к узлу назначения, то он отсылает соответствующее сообщение узлу-отправителю. Эти функции выполняет протокол ICMP – Internet Control Message Protocol, описание которого приведено в документе RFC 792. Хотя его сообщения инкапсулируются в IP-пакет, протокол ICMP является протоколом сетевого уровня. Рассматриваемый протокол обеспечивает взаимодействие между программным обеспечением протокола IP, функционирующим на разных сетевых узлах. Однако, протокол не способен информировать промежуточные узлы о возникших ошибках, поскольку в IP-пакете нет поля для записи маршрута. Соответственно, когда пакет пришел на некий маршрутизатор и в ходе его обработки обнаружилась необходимость отправки сообщения ICMP, то единственным получателем такого сообщения будет узел – отправитель исходного пакета.

Протокол ICMP генерируют два вида сообщений: управляющие и сообщения об ошибках, но он не содержит никаких средств исправления ошибок; эта задача решается программным обеспечением узла-отправителя.

Сообщения ICMP начинаются тремя обязательными полями: «Тип», «Код» и «Контрольная сумма». Кроме этого, большинство сообщений включают в себя заголовок и первые 64 бита пакета, в котором была обнаружена ошибка, сообщение о которой несет данный пакет ICMP. Это сделано для более точной идентификации источника ошибки на узле-отправителе. Поле «Тип» определяет содержание сообщения и его формат. На слайде приведены некоторые значения этого поля.

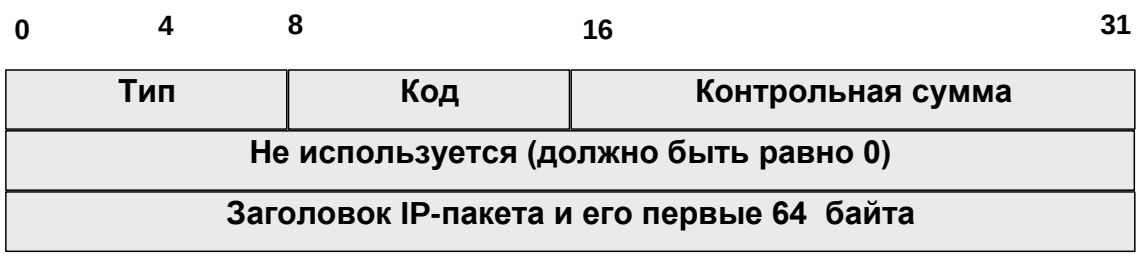
Сообщения «Запрос эха» и «Ответ на эхо» являются одними из самых используемых (команда **ping**). Поскольку эти сообщения передаются в IP-пакетах, то успешный прием ответа свидетельствует о работоспособности основных компонентов сетевой транспортной системы, т.е. успешной

маршрутизации, работоспособности IP-модулей стека протоколов на станциях отправителе и получателе. На рисунке внизу слайда представлен их формат.

Поля «Идентификатор» и «Последовательный номер» используются отправителем для проверки соответствия между ответом и запросом. Поле «Необязательные данные» имеет переменную длину и содержит данные, которые необходимо вернуть отправителю.

ICMP – протокол обмена управляющей информацией

Значение «Код»	Причины	Значение «Код»	Причины
0	Сеть недостижима	5	Ошибка при маршрутизации от источника
1	Устройство недостижимо	6	Сеть назначения неизвестна
2	Протокол недостижим	11	Сеть недостижима из-за класса обслуживания
3	Порт недостижим	12	Устройство недостижимо из-за класса обслуживания
4	Требуется фрагментация		



Формат сообщения «Получатель недостижим»

14

Сообщение «Получатель недостижим» генерируется маршрутизатором, в ситуации, когда он не может доставить пакет по назначению. Это сообщение содержит поле «Код», которое уточняет причину невозможности доставки пакета. Некоторые из них представлены в таблице.

Сообщение «Изменение маршрута» (***Слайд 13***) отправляется маршрутизатором отправителю, находящемуся с ним в одной физической сети, и свидетельствует об использовании последним неоптимального маршрута к узлу-назначения. Это сообщение содержит адрес маршрутизатора, использование которого является предпочтительным. Таким образом, начиная работу, отправитель может знать лишь один маршрут. В дальнейшем, его таблица маршрутизации, посредством этих сообщений ICMP, будет дополнена более точной информацией об «оптимальных» маршрутах. Весь этот механизм позволяет узлу отправителю минимизировать размер своей таблицы маршрутов.

Обработка IP-пакетов маршрутизатором.

Можно выделить две основные функции маршрутизатора:

1. определение оптимального маршрута пакета и
2. коммутацию пакетов с одного физического интерфейса на другой.

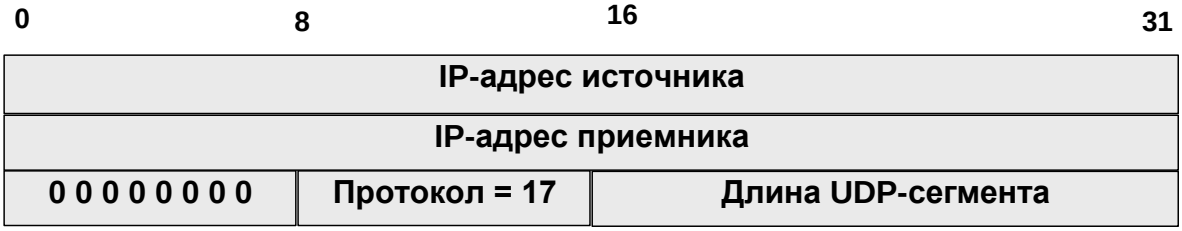
Определение маршрута, как правило, реализуется программным образом, а передача (коммутация) пакетов с одного интерфейса на другой в современных маршрутизаторах выполняется аппаратно. Программное обеспечение маршрутизатора включают в себя набор протоколов маршрутизации, процедуры управления базой данных (маршрутной таблицей) и процедуры поддержки определенных сервисов (фрагментация, фильтрация и т.п.).

Протокол UDP.

Два протокола транспортного уровня, UDP и TCP, обеспечивают IP-сетям механизмы взаимодействия прикладных процессов, выполняющихся на конечных станциях. Напомним, что собственно протокол IP «умеет» доставлять пакеты данных взаимодействующим хостам, но не «знает» как обеспечить взаимосвязь приложений и не имеет почти никаких средств обеспечения надежности доставки сообщений - он проверяет лишь целостность заголовка пакета.

Протокол UDP

Формат UDP-сегмента



Формат псевдозаголовка UDP-сегмента

15

Протокол UDP (User Datagram Protocol) описан в документе RFC 768. Он ориентирован на сервис без установления соединений и не обеспечивает надежную передачу сегментов между сетевыми приложениями. Это очень простой протокол, который развивает возможности IP-протокола лишь в части демультиплексирования потока пакетов по признаку принадлежности их определенному приложению и контроля целостности данных. Взаимодействие между прикладными процессами UDP реализует посредством механизма **протокольных портов**. Протокольный порт можно определить как абстрактную точку присутствия конкретной прикладной программы, выполняющейся на конкретном хосте. Когда рабочая станция получает пакет, в котором указан ее IP-адрес, она может направить его определенной программе, используя уникальный номер порта, назначенный этой программе в ходе выполнения процедуры установления соединения. Таким образом, в стеке протоколов TCP/IP порт

является механизмом поддержания рабочей станцией одновременного выполнения нескольких прикладных процессов.

Каждый порт (прикладной процесс) идентифицируется целым положительным числом (номером порта). Номера портов приложения, выполняющегося на разных станциях, указываются в заголовке UDP-сегмента. Эта информация дополняется на сетевом уровне IP-адресами взаимодействующих станций. Благодаря этому, создается видимость непосредственного обмена данными между процессами.

Сегмент данных протокола UDP (иногда его называют пользовательской дейтограммой) состоит из двух частей: заголовка и области данных (см. слайд). Заголовок имеет четыре 16-битных поля, определяющих порт отправителя, порт получателя, длину сегмента и контрольную сумму.

Поле «Длина UDP-сегмента» содержит количество байтов в дейтограмме с учетом длины ее заголовка.

Вычисление контрольной суммы дейтограммы UDP является опциональным. При работе в надежных локальных сетях она не вычисляется и тогда это поле заполняется нулями. Процедура подсчета контрольной суммы содержит две особенности. Первая состоит в дополнении дейтограммы нулевыми битами до размера, кратного 16. Это делается только на время вычисления контрольной суммы, и незначащие нули не передаются. Второй особенностью является дополнение, на период подсчета контрольной суммы, заголовка сегмента **псевдозаголовком.** Его формат представлен на рисунке внизу.

Псевдозаголовок включается перед заголовком дейтограммы; он имеет длину 12 байтов; поле «Протокол» содержит тип протокола сетевого уровня (IP, ICMP); его значение, как и значения полей с IP-адресами, должны быть извлечены из заголовка IP пакета. Поле «Контрольная сумма» на время ее вычисления заполняется нулями.

Такое дополнение дейтограммы выполняется как на передающей, так и на приемной станции и оно служит гарантией, что если контрольные суммы совпали, то дейтограмма достигла нужной станции и нужного порта. Еще раз подчеркну, что псевдозаголовок и дополнение нулями не передаются.

Если контрольные сумма, вычисленная приемником, не совпала с контрольной суммой, указанной в дейтограмме, то UDP-сегмент уничтожается и никаких уведомлений передающей станции об этом не передается.

Протокол TCP.

Протоколы TCP и IP являются «рабочими лошадками» Интернет. В этом разделе будут рассмотрены механизмы, посредством которых TCP обеспечивает надежный, ориентированный на установление соединений потоковый сервис в дейтограмной сети IP. В число этих механизмов входит вариант алгоритма ARQ с выборочным повторением и алгоритм управления перегрузками на базе индикации потерь сегментов. Подробно протокол описан в документах RFC 793 и RFC 1122.

Надежный потоковый сервис.

Задачей протокола TCP является организация и поддержание логического полнодуплексного соединения между двумя приложениями, выполняющимися на конечных станциях, соединенных ненадежной дейтограмной сетью, которое обеспечивает обмен упорядоченным потоком байтов и управление его интенсивностью. Дуплексность соединения позволяет вести передачу данных в одном из направлений и после того, как соединение противоположного направления было закрыто.

Аналогично UDP, каждый прикладной процесс для TCP-модуля представляется номером порта. Структура из пары переменных (порт, IP-адрес) называется **сокетом.** Соединение между отправителем и получателем однозначно определяется двумя сокетами. Для хранения всей информации, необходимой для установления и поддержания соединения, определена специальная структура данных – **блок управления передачей (Transmission Control Block - TCB)**. В эту структуру, кроме двух сокетов, входят флаги безопасности и приоритета соединения, указатели буферов отправителя и получателя, указатели номеров очередного сегмента и сегмента повторной посылки, а также ряд других переменных.

TCP не сохраняет границы сообщений и рассматривает данные, которые поступают ему от приложения, как поток байтов. Свои сегменты он формирует так, как считает необходимым, но с учетом свойств протокола сетевого уровня (сегмент должен полностью поместиться в IP-пакет). Таким

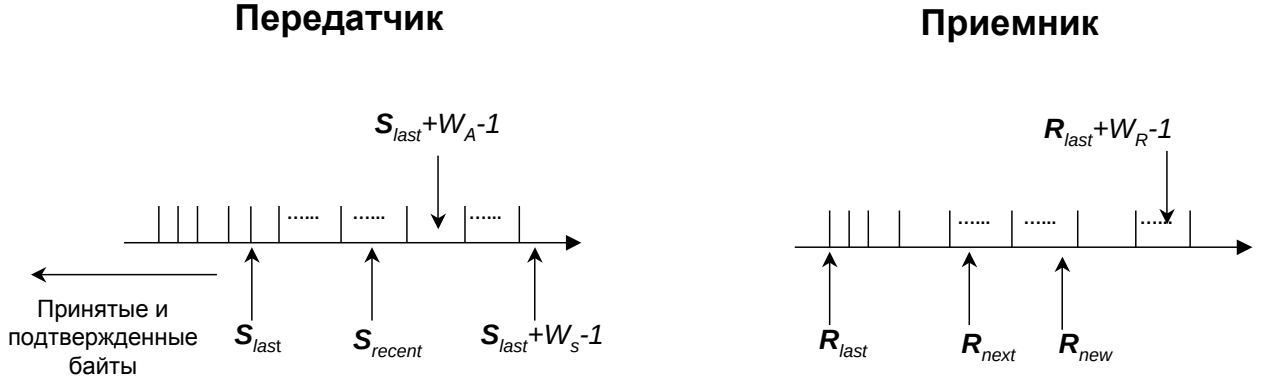
образом, если приложение отсылает сообщение, размер которого составляет 1000 байтов, то на приемной стороне оно может быть представлено двумя частями по 500 байт, тремя частями по 300, 300 и 400 байт и.т.д.

Адаптационные механизмы протокола.

Надежный потоковый сервис протокол TCP обеспечивает посредством специальных адаптационных механизмов. Напомним, что этот протокол предполагает наличие сетевой среды, в которой пакеты могут теряться, дублироваться, приходить в узел назначения с нарушением порядка их отправки. Для идентификации таких нарушений в протоколе введена последовательная нумерация **байтов** сегмента. Порядковый номер первого байта сегмента передается в его заголовке и он считается порядковым номером сегмента. Номер первого байта для каждого **соединения** выбирается случайным образом в период установления соединения. Поскольку каждый байт сегмента оказывается пронумерованным, то легко решается задача опознания на приемной стороне нарушений порядка их доставки. Механизм подтверждения приема носит накопительный характер, т.е. подтверждение приема байта с номером N, означает, что байты с номерами N-1, N-2, … успешно получены. Диапазон возможных номеров байтов ограничен числами от 0 до 2^{31}-1. Заметим, что при скорости передачи в 100 Мбит/с полный цикл использования всего допустимого диапазона номеров составляет 5,4 минуты, что вполне достаточно для того, чтобы любой сегмент был получен приемной станцией, а все его дубли были уничтожены.

Протокол TCP

Slast **- указатель наименьшего номера неподтвержденных байтов**
Sresent **- указатель номера последнего из отправленных байтов**
Slast+Ws–1 **– указатель наибольшего номера байта**
который передающий модуль может принять от приложения (но не отправить в сеть)
Rlast **- указатель наименьшего номера байта, который еще не был передан приложению**
Rnext **- указатель номера следующего ожидаемого к приему байта**
Rnew **- указатель наибольшего номера корректно принятого байта.**



Окно передачи и приема в TCP-модулях

Передача данных в протоколе TCP регулируется на основе алгоритма скользящего окна.

Окно передачи определяется тремя указателями:

- S_{last} - указатель наименьшего номера неподтвержденных байтов,
- S_{resent} - указатель номера последнего из отправленных байтов,

- $S_{last}+W_s-1$ – указатель наибольшего номера байта, который передающий модуль может принять от приложения (но не отправить в сеть).

Заметим, что передатчик отправит сегмент в сеть лишь после получения от приложения некоторого числа байтов, превышающего заранее установленный порог, либо после истечения определенного интервала времени. При этом, величина сегмента должна быть такой, что наибольший номер его байта, не должно превосходить величину $S_{last}+W_A-1$, где W_A - это так называемое объявляемое окно, параметр определяемый размером свободного буферного пространства приемного приложения.

Величина окна передачи (W_s) является механизмом адаптации интенсивности передатчика к требованиям сети и изменяется под воздействием целого ряда факторов.

В свою очередь, приемный модуль протокола TCP формирует свое окно, которое также управляется тремя параметрами:

- R_{last} - указатель наименьшего номера байта, который еще не был передан приложению
- R_{next} - указатель номера следующего ожидаемого к приему байта,
- R_{new} - указатель наибольшего номера корректно принятого байта.

Отметим, что R_{new} может оказаться больше R_{next}. Такая ситуация является следствием того, что приемный модуль не отбрасывает байты, принятые в пакетах, не содержащих ошибок, но пришедших с нарушением порядка их следования. Если размер буферной памяти приемного модуля равен W_R байт (максимальный размер приемного окна), то всегда существует возможность принять байты с номерами, не превышающими $R_{last}+W_R-1$. При этом, мы предполагаем, что приложение не обязательно читает байты немедленно после их приема.

Прибывший сегмент проверяется на целостность данных и при положительном результате такой проверки его байты, если их число не превышает размер W_R, записываются в соответствующую область памяти приемного буфера. Если принятый сегмент имеет номер R_{next}, то значение этого указателя обновляется, а передающему приложению отсылается положительное подтверждение с новым номером R_{next}., что свидетельствует об успешном приеме всех сегментов с порядковыми номерами меньшими R_{next}. Это подтверждение используется передающим модулем для обновления значения своего указателя S_{last} и, тем самым, передающее окно перемещается вперед, позволяя отправлять сегменты с более высокими порядковыми номерами.

TCP располагает механизмом **управления потоком**, предотвращающим переполнение приемного буфера. Этот регулятор представляет собой, так называемое, **объявляемое окно (advertised window) –** специальное поле в заголовке сегмента, передаваемого от приемника к передающему узлу. Это поле информирует передающий модуль о доступной величине буферной памяти приемника. Величина объявляемого окна определяется выражением

$$W_A = W_R - (R_{new} - R_{last}).$$

Получив значение W_A, передающий модуль может из накопленных в его буфере данных сформировать сегмент, величина которого не превышает W_A, т.е.

$$S_{recent} - S_{last} \le W_A.$$

Для иллюстрации эффекта регулирования интенсивности потока, генерируемого приложением, рассмотрим ситуацию, когда приложение на приемной стороне прекращает читать данные из буфера своего модуля TCP. Ясно, что величина R_{new} будет увеличиваться, а R_{last} оставаться постоянной; такая динамика приводит к постоянному уменьшению величины объявляемого окна и передающий модуль TCP вынужден уменьшать размер отправляемых сегментов. В конце концов, W_A станет равным нулю и передающий модуль TCP прекратит отправку данных. Однако, запись данных приложения в буфер передающего модуля будет продолжаться до тех пор, пока он не примет W_s байт. В этот момент работа приложения будет полностью заблокирована.

Надежность передачи данных обеспечивается применением алгоритма повторной передачи с выборочным повторением. Когда протокол станции-отправителя передает сегмент с номером *M,* он помещает его копию в очередь повторной передачи и запускает специальный таймер повторной передачи; при получении подтверждения о благополучном приеме сегмента *M*, его копия уничтожается. В противном случае, после истечении таймера повторной передачи модуль TCP снова отправит этот сегмент в сеть. Время доставки сегментов в сетевой среде, в которой работает протокол TCP (Интернет,

например), может изменяться в значительных пределах даже в течении одного соединения. Поэтому выбор значения таймера повторной передачи представляет собой непростую задачу. В протоколе TCP для этого предусмотрена адаптивная процедура, базирующаяся на постоянном измерении интервала времени от посылки сегмента до получения квитанции о его приеме (τ_n). Эти значения непрерывно усредняются с некими весовыми коэффициентами, уменьшающимися от последнего к предыдущему замером. В результате получают значение параметра **«полное время оборота»** (**round-trip time**, t_{RTT}).

$t_{RTT}(new) = \alpha t_{RTT}(old) + (1-\alpha)\tau_n$,

где $0 < \alpha \le 1$, типичное значение $\alpha = 7/8$. Значение таймера повторной передачи (t_{out}) учитывает также дисперсию значения t_{RTT}:

$t_{out} = t_{RTT} + k\sigma_{RTT}$,

где k некоторая константа. Таким образом, если девиация значений полного времени оборота будет значительная, то и значения таймера повторной передачи будут увеличены в сравнении со средним значением t_{RTT}. Поскольку вычисление значения стандартного уклонения относительно трудоемко, то в практически используемых процедурах оценка вариации значений t_{RTT} базируется на вычислении абсолютного уклонения $|\tau_n - t_{RTT}|$:

$t_{out} = t_{RTT} + 4d_{RTT}$,

$d_{RTT}(new) = \beta d_{RTT}(old) + (1-\beta)|\tau_n - t_{RTT}|$,

при типичном значении $\beta = 0.25$.

Функционирование протокола TCP.

Протокол TCP

Формат TCP-сегмента

В этом разделе будут рассмотрены структура сегмента TCP, механизмы установления соединения, передачи данных и ликвидации соединения. Формат TCP сегмента представлен на слайде. Его заголовок содержит 20-байтную фиксированную часть и опциональную часть переменной длины.

«Порт источника» и **«Порт назначения» -** определяют передающее и приемное приложения, соответственно.

«Порядковый номер сегмента» - определяет позицию первого байта данных сегмента в байтовом потоке источника при значении флага SYN=0 (в режиме передачи данных). Напомним, что TCP нумерует байты, а не сегменты и если порядковый номер текущего сегмента равен 567, а поле данных содержит 12 байт, то следующий сегмент будет иметь порядковый номер 579. В режиме установления соединения, когда флаг SYN установлен в 1, в этом поле содержится начальный номер последовательности номеров байтов данного потока **(ISN – initial sequence number**); значение номера первого байта данных этого потока будет ISN+1. Отметим также, что соединения TCP являются дуплексными и в каждом из направлений передачи устанавливается своя нумерация.

«Порядковый номер подтверждения» - это поле в режиме с установленным флагом ACK (режим передачи данных) содержит порядковый номер байта данных, который передающий модуль ожидает получить от приемного узла; тем самым подтверждается правильность приема всех предыдущих байтов. В режиме установления соединения (ACK=0) значение этого поля не учитывается.

«Смещение данных» - поле определяет длину заголовка сегмента в 32-битных словах; эта информация позволяет приемному модулю определить начало поля данных, т.к. заголовок может содержать опциональное поле переменной длины.

«Резерв» - поле в настоящее время не используется и заполняется нулями.

«Контрольные биты» - поле длиной 6 бит, каждый из которых является флагом; их последовательность и смысл следующие:

URG – флаг срочности передачи сегмента

ACK – флаг указывающий на достоверность значений в поле **«Порядковый номер подтверждения»**

PSH – включена функция «проталкивания» сегмента, т.е. модуль TCP должен передать сегмент приложению немедленно

RST – указание приемному модулю разорвать соединение по причине каких-то аномалий; используется для перезагрузки соединения

SYN – флаг установления соединения, синхронизации порядковых номеров сегментов

FIN – флаг, индицирующий, что у передающего модуля нет данных для передачи; передающее приложение остается в соединении с приемным и принимает данные последнего.

«Размер окна» - поле опрделяет количество байтов, которое модуль TCP может принять *(W_a)*.

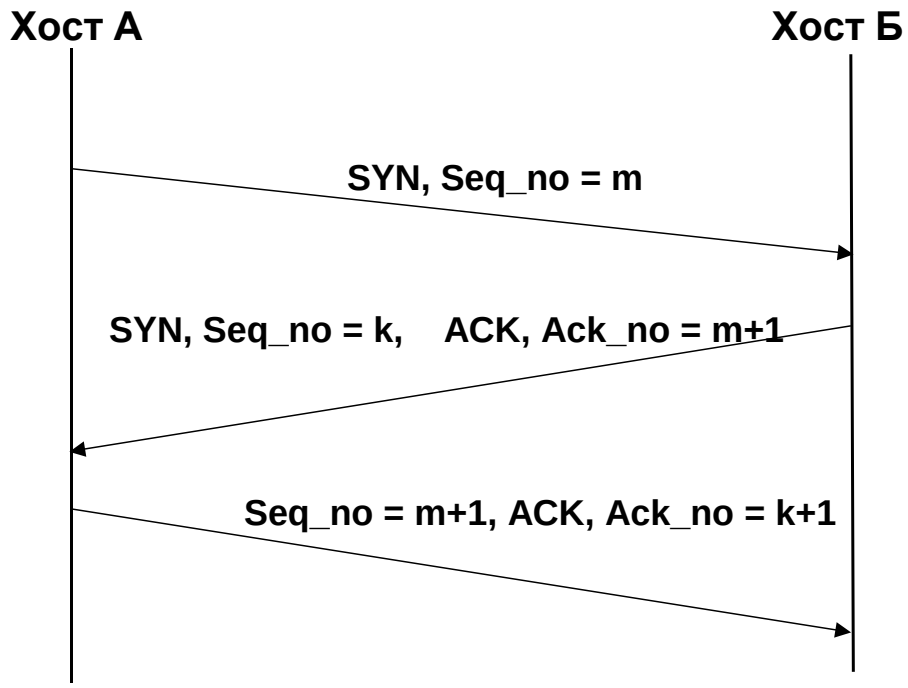
«Контрольная сумма» - значение этого поля рассчитывается по всему сегменту с дополнением его нулями до размера кратного 16 битам и 96 битным псевдозаголовком, включаемым перед заголовком TCP и содержащим сетевые адреса отправителя и получателя, тип протокола и длину TCP сегмента. Эти дополнения используются только для расчета контрольной суммы и не передаются.

«Указатель срочности» - значение этого поля при установленном флаге URG, будучи добавленным к значению поля **«Порядковый номер сегмента»,** определяет последний байт срочных данных. Поскольку приемный модуль TCP передает приложению байты строго по порядку, то все байты, содержащиеся в приемном буфере, вплоть до байта с определенным, как указано выше номером, будут рассматриваться как срочные.

«Опции» - поле используется для определения других, не предусмотренных заголовком, функций. Так например, это поле часто используется для определения **максимального размера сегмента** (maximum segment size - MSS). При использовании протокола в высокоскоростных сетях это поле используется для задания таких параметров как **«Коэффициент масштабирования окна»** (до 2^{14}) и **«Временная метка».** Последние важны в ситуации когда полный цикл нумерации байт может быть пройден за время существования соединения. Наличие временных меток в каждом сегменте позволяет также вычислить время полного оборота (*RTT*).

Теперь рассмотрим работу протокола в различных фазах жизни соединения.

Протокол TCP



Процедура установления TCP-соединения

18

Фаза установления соединения.

Фаза установления соединения, предшествующая фазе передачи данных, содержит следующие действия.

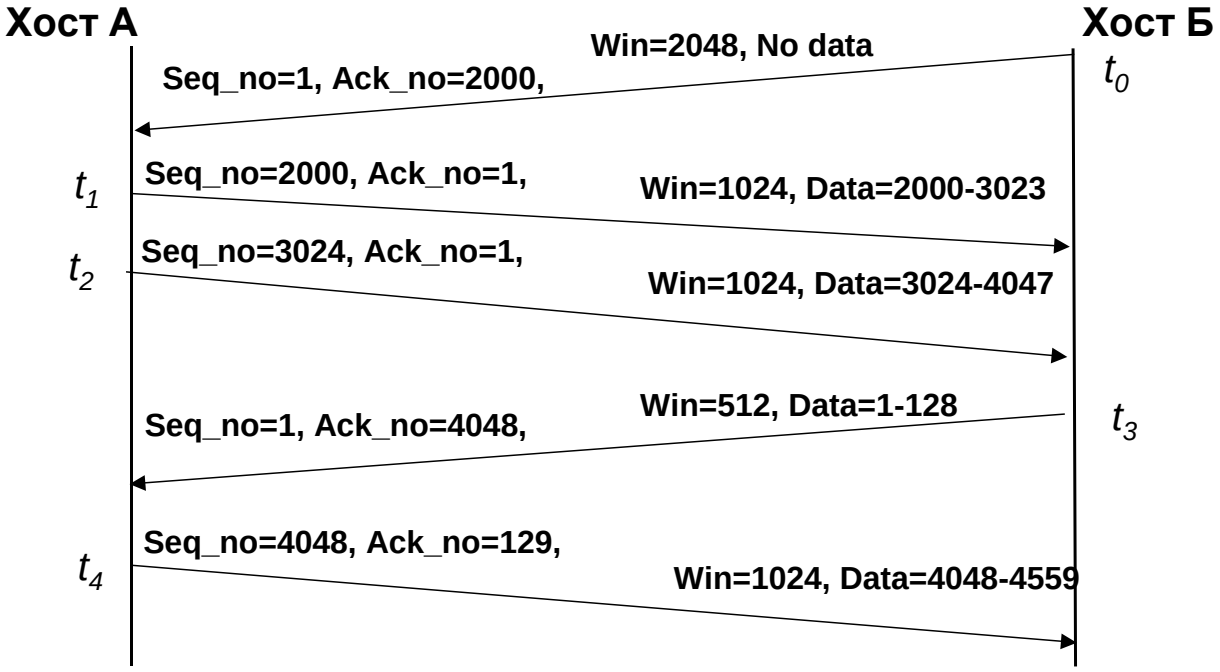
1. Хост **А** отправляет хосту **Б** запрос соединения посредством установки флага SYN и инициализирует значение начального номера нумерующей последовательности (Seq_no = m).
2. Хост **Б** отвечает на этот запрос установкой флага ACK и определяет поле «**Порядковый номер подтверждения»** значением на единицу большим m (Ack_no = m+1); одновременно, хост **Б** в своем ответе **А** отправляет запрос соединения (SYN) и также инициализирует значение начального номера своей нумерующей последовательности (Seq_no = k).
3. Хост **А** отвечает на запрос соединения от хоста **Б** установкой флага ACK и подтверждением ожидания следующего байта данных с порядковым номером k+1 (Ack_no = k+1); при этом, значение поля «**Порядковый номер сегмента»** устанавливается в значение m+1 (Seq_no = m+1).

Такая трехэтапная процедура установления соединения гарантирует согласование начальных значений нумерующих последовательностей взаимодействующих TCP-модулей, что принципиально важно для последующего функционирования соединения. Случайный характер выбора начальных значений Seq_no является достаточно надежной мерой, предупреждающей установление на обоих концах соединения одинаковых начальных номеров.

Заметим, что в фазе установления соединения каждый SYN-сегмент может содержать опциональные параметры и любой хост может отказать в соединении посредством отсылки сегмента с установленным флагом RST.

Протокол TCP поддерживает два типа соединения – активное и пассивное. Так, в приложениях, построенных по клиент-серверной архитектуре, сервер выполняет пассивное соединение (формирует свой сокет и переходит в режим «прослушивания»), сообщая тем самым своему модулю TCP о готовности принять запрос соединения. Когда клиент желает установить связь с сервером, он выполняет процедуру активного соединения. В нее входит создание сокета на клиентской стороне и выполнение описанных выше процедур TCP-соединения.

Протокол TCP



Передача данных по TCP-соединению

19

Фаза передачи данных.

Предоставление приложениям сервиса надежной доставки данных в протоколе TCP обеспечивается использованием алгоритма ARQ с выборочным повторением и механизма скользящего окна. При этом, особенностью протокола TCP является реализация скользящего окна не на уровне сегментов, а на уровне байтов. Протокол также обеспечивает управление потоком в фазе передачи данных посредством регулирования величины объявляемого окна и величины окна передачи. Слайд иллюстрирует процесс управления интенсивностью потока.

Пусть в момент t_0 TCP-модуль хоста **В** объявил величину своего окна равной 2048 байт и номер следующего ожидаемого байта 2000. Такой размер окна позволяет хосту **А** отправить без подтверждения 2 Кбайта данных, однако в его выходном буфере имеется лишь 1024 байта данных. Хост **А** отправляет эти данные нумеруя байты начиная с 2000. Одновременно, он объявляет величину своего окна равной 1024 байта и подтверждает, что номер ожидаемого первого байта от хоста **Б** должен быть равен 1. Хост **Б** задерживает выдачу подтверждения на прибывший сегмент данных, полагая, что у него появятся данные для отправки хосту **А,** вместе с которыми он отправит и подтверждение. Тем временем, в момент t_2 модуль TCP хоста **А** снова получил от своего приложения 1024 байт данных и передал их хосту **Б**. После этого величина окна отправки на хосте **А** стала равной нулю и дальнейшая отправка им данных до получения подтверждения от хоста **Б** оказывается невозможной. В момент t_3

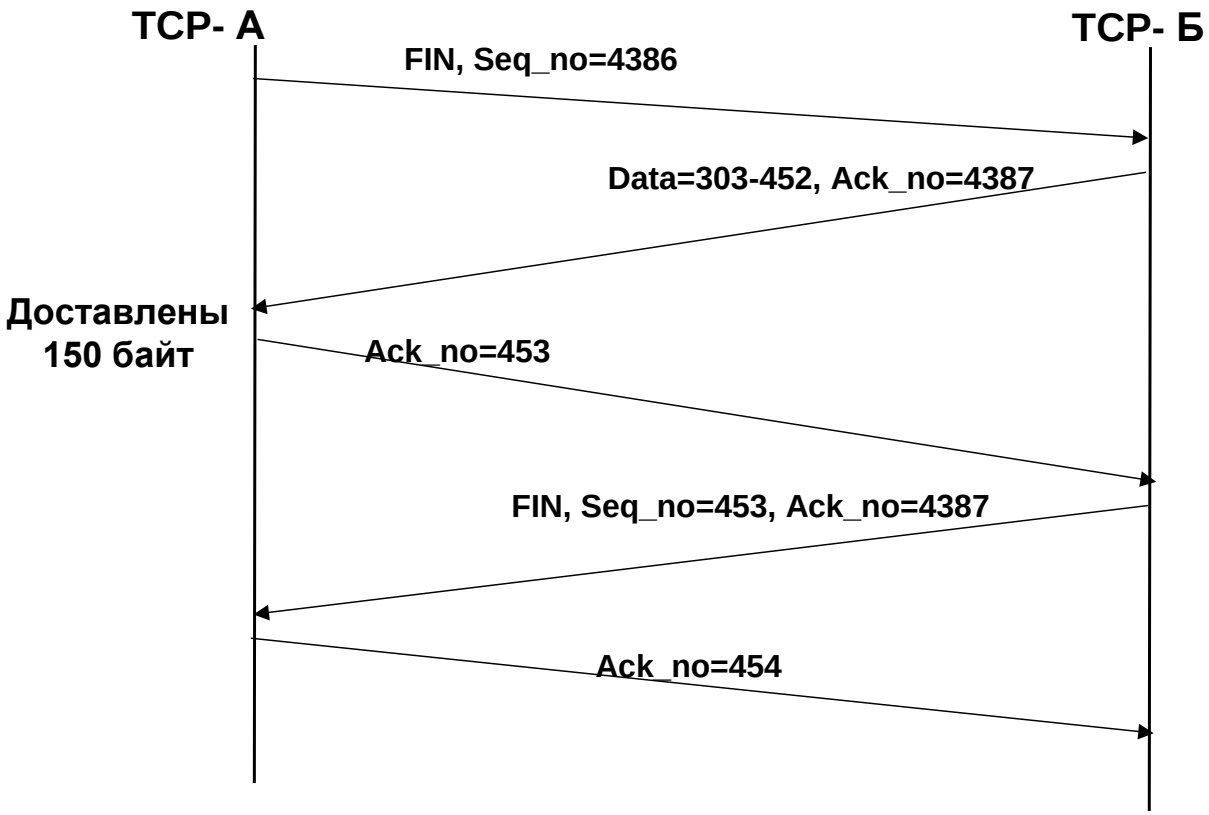
модуль TCP хоста **Б** получил 128 байт данных для отправки; вместе с ними он отправляет подтверждение получения от хоста **А** двух сегментов данных, указывая Ack_no=4048. К этому моменту в буферной памяти модуля TCP хоста **Б** оказывается свободными лишь 512 байт, поэтому он объявляет величину своего окна приема равной 512. Когда хост **А** получит этот сегмент, он установит величину окна отсылки равной 512 байт и, несмотря на то, что в момент t_4 в его буфере имеется 2048 байт данных, он сможет отослать только 512 байт и не перегрузит буфер хоста **Б.**

Предыдущее рассмотрение показывает, что задержка отправки подтверждения позволяет более экономно использовать пропускную способность канала. Это особенно актуально в ситуациях, подобных login-сессиям, когда пользователь вводит свои аутентификационные данные по одному символу. Пересылка каждого из них, сама по себе весьма затратна (на один байт информации приходится не менее 40 байтов заголовков TCP и IP уровней), а выдача подтверждения на каждый такой сегмент еще более усугубляет ситуацию. Решение, экономящее пропускную способность канала связи, было предложено Нейглом (Nagle). Идея алгоритма достаточно проста. Когда интерактивное приложение требует отсылки символа, модуль TCP отправляет его и ждет подтверждения. Поступающие в этот период времени от приложения символы не передаются, а буферизируются; они отправляются все вместе в одном сегменте после получения подтверждения на первый отправленный символ. В локальных сетях, где задержки доставки сегментов относительно малы и каналы связи являются достаточно широкополосными, применение рассмотренного алгоритма является нецелесообразным.

Существует еще одна ситуация, в которой механизм регулирования окна протокола TCP может приводить к неэффективному использованию полосы пропускания. Пусть передающее приложение имеет большой объем данных для передачи, а принимающее приложение может читать буфер своего TCP модуля лишь небольшими порциями. Ясно, что, в конце концов, это приведет к объявлению приемным модулем малого значения окна; соответственно, передающий модуль будет формировать сегменты с малым объемом данных, что и увеличит «накладные» расходы протокола при их передаче. Это явление часто называют «синдромом ленивого окна». Для уменьшения его негативного влияния ограничивают возможность объявлять малые значения приемного окна. А именно, пока размер свободного буферного пространства приемника не достигнет половины его объема, либо половины размера максимально допустимого сегмента, размер приемного окна не объявляется (т.е. считается равным нулю). Данные приложения на передающей стороне буферизируются и отправляются после объявления ненулевого приемного окна уже достаточно большими сегментами.

Рассмотрим еще проблему зацикливания нумерующей последовательности, характерную для работы протокола TCP в высокоскоростных средах. Исходная спецификация протокола предполагала, что максимальное время жизни сегмента составляет 2 минуты. Последовательность из 2^{32} номеров способна пронумеровать 4294967296 байт. Для их передачи по линии с пропускной способностью 2048 Кбит/с потребуется $(2^{32} \times 8)/(2{,}048 \times 10^6) = 4{,}5$ часа, а по линии OC-48 (2,4 Гбит/с) всего 14 секунд. Ясно, что на современных высокоскоростных линиях протокол TCP может терять работоспособность. Решение этой проблемы было найдено посредством определения в опциональном поле заголовка сегмента 4-х байтовой временной метки. Приемный модуль включает ее в свой ACK-сегмент, и таким образом объединяются два способа разметки последовательности отправляемых сегментов. Это увеличивает период нумерующей последовательности до 2^{64}. Ясно, что для реализации этого механизма таймер временных меток должен изменять свое значение, по крайней мере, один раз за время необходимое для отправки 2^{31} байт (максимальное значение окна передачи). Это требование определяет нижнюю границу частоты этого таймера. Эта частота не должна быть и настолько высокой, чтобы полный цикл показаний таймера был пройден за время, меньшее максимального времени жизни сегмента. Значения в диапазоне от 1 Гц до 1 кГц удовлетворяют этим ограничениям. Например, при частоте в 1кГц протокол будет успешно работать на линиях с пропускной способностью до 8 Тбит/с. [RFC 1323].

Протокол TCP



Фаза ликвидации соединения.

Протокол TCP реализует процедуру поэтапной ликвидации соединения, предполагающую независимое его закрытие в обоих направлениях. Необходимость в закрытии соединения возникает, когда приложение сообщает своему модулю TCP об отсутствии у него данных для отправки. TCP модуль завершает передачу данных, находящихся в его буфере, ожидает получения подтверждения об их успешном приеме и отправляет приемному модулю сегмент с установленным флагом FIN. Получив этот сегмент, приемный модуль информирует свое приложение о завершении поступления данных от передающего приложения, но продолжает отсылать данные (если они есть) в противоположном направлении. Получив подтверждение на отправленные данные, модуль TCP отправляется сегмент FIN в противоположном направлении и, после получения на него подтверждения ACK, соединение считается ликвидированным.

Хост **А** инициализирует процедуру разрыва соединения, отправляя сегмент с флагом FIN. Модуль TCP хоста **Б** подтверждает прием этого сегмента и передает извещение о запросе на закрытие соединения своему приложению. Одновременно, располагая данными для хоста **А,** модуль TCP хоста **Б** отправляет сегмент со 150 байтами данных хосту **А**, и получает подтверждение их приема. Получив от своего приложения подтверждение разрыва соединения протокольный модуль хоста **Б** отправляет встречный сегмент FIN и получает на него подтверждение. Модуль TCP хоста **А** переходит в состояние ожидания и запускает таймер TIME_WAIT с начальным значением задержки равным удвоенному максимальному времени жизни сегмента. В этот период единственным сегментом, который может прийти на хост **А**, является повторный сегмент FIN от хоста **Б** (если соответствующий сегмент ACK от хоста **А** был утерян). Если такой сегмент приходит, то хост **А** повторно отсылает сегмент ACK и вновь перезапускает таймер TIME_WAIT. При достижении этим таймером значения нуль, хост **А** ликвидирует соединение и удаляет запись о нем из таблицы соединений.

Состояние ожидания обеспечивает выполнение еще одной задачи, а именно, оно защищает будущие реализации соединения между этими же прикладными процессами от обработки задержавшихся в сети сегментов предыдущего соединения. За двойное время жизни все, не доставленные сегменты этого соединения, будут уничтожены.

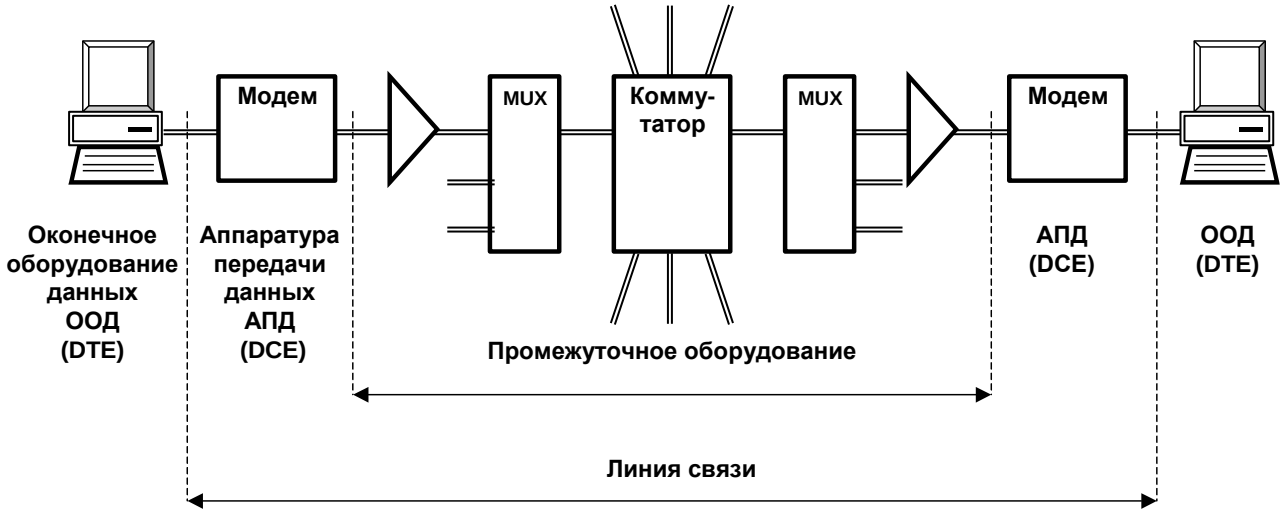
Протокол TCP располагает также механизмом срочной ликвидации соединения посредством отправки RST-сегмента. Отправка этого сегмента является на получение сегмента, адресованного приложению, которого на данном порте нет. Передающий модуль TCP, получив сегмент RST, уничтожает все данные, находящиеся в его буфере. Приемный модуль TCP, получив сегмент RST, информирует о ликвидации соединения соответствующий прикладной процесс.

Лекция 4. Основы передачи дискретных данных.

Любая сетевая технология должна обеспечить надежную и быструю передачу дискретных данных на физическом уровне по линиям связи. И все сетевые технологии при всех их отличиях базируются на общих принципах передачи дискретных данных. Эти принципы находят свое воплощение в:

- методах представления двоичных сигналов (т.е. нулей и единиц) с помощью импульсных или синусоидальных сигналов в линиях связи различной физической природы,
- методах обнаружения и коррекции ошибок,
- методах компрессии и
- методах коммутации.

Линии связи



Состав линии связи

2

Линии связи.

Линия или канал связи состоит в общем случае из:

- физической среды (media), по которой передаются электрические сигналы,
- аппаратуры передачи данных,
- промежуточной аппаратуры.

В зависимости от среды линии делятся на:

- проводные или воздушные,
- кабельные (медные и волоконно – оптические),
- радиоканалы наземной и спутниковой связи.

Проводные (воздушные) линии - провода без каких-либо изолирующих или экранирующих оплеток, проложенные между столбами и висящие в воздухе. Традиционно передают телефонные или телеграфные сигналы, но при отсутствии других возможностей эти линии используются и для передачи компьютерных данных. Скоростные качества и помехозащищенность плохие. Сегодня быстро вытесняются кабельными.

Кабельные линии - проводники, заключенных в несколько слоев изоляции: электрической, электромагнитной, механической, а также, возможно, климатической. Три основных типа кабеля:

- кабели на основе скрученных пар медных проводов,
- коаксиальные кабели с медной жилой,
- волоконно-оптические кабели.

Витая пара существует в экранированном варианте (Shielded Twisted Pair, STP), когда пара медных проводов обертывается в изоляционный экран, и неэкранированном (Unshielded TwistedPair, UTP), когда изоляционная обертка отсутствует. Скручивание проводов снижает влияние внешних помех на сигналы.

Коаксиальный кабель - несимметричная конструкция и состоит из внутренней медной жилы и оплетки, отделенной от жилы слоем изоляции. Существует несколько типов коаксиального кабеля — для локальных сетей, для глобальных сетей, для кабельного телевидения и т. п.

Волоконно-оптический кабель состоит из тонких (5-60 микрон) волокон, по которым распространяются световые сигналы. Это наиболее качественный тип кабеля — он обеспечивает передачу данных со скоростью до 10 Гбит/с и выше и к тому же лучше других типов обеспечивает защиту данных от внешних помех.

Радиоканалы наземной и спутниковой связи образуются с помощью передатчика и приемника радиоволн. Это могут быть каналы в диапазоне коротких, средних и длинных волн (КВ, СВ и ДВ), называемые также диапазонами амплитудной модуляции. Обеспечивают дальнюю связь, но при невысокой скорости передачи данных.

УКВ, для которых характерна частотная модуляция (Frequency Modulation, FM), а также диапазонах сверхвысоких частот (СВЧ или microwaves). В диапазоне СВЧ (свыше 4 ГГц) сигналы уже не отражаются ионосферой Земли и для устойчивой связи требуется наличие прямой видимости между передатчиком и приемником. Поэтому такие частоты используют либо спутниковые каналы, либо радиорелейные каналы, где это условие выполняется.

Аппаратура линий связи

Аппаратура передачи данных (DCE) непосредственно связывает компьютеры или локальные сети пользователя с линией связи и является **пограничным** оборудованием. Традиционно аппаратуру передачи данных включают в состав линии связи. Примеры DCE - модемы, терминальные адаптеры сетей ISDN, оптические модемы, устройства подключения к цифровым каналам. DCE работает на

физическом уровне, отвечая за передачу и прием сигнала нужной формы и мощности в физическую среду.

Оконечное оборудование данных (DTE). Примеры DTE - компьютеры или маршрутизаторы локальных сетей. Эту аппаратуру не включают в состав линии связи.

Разделение оборудования на классы DCE и DTE в локальных сетях является достаточно условным. Например, адаптер локальной сети можно считать как принадлежностью компьютера, то есть DTE, так и составной частью канала связи, то есть DCE.

Промежуточная аппаратура обычно используется на линиях связи большой протяженности и решает две основные задачи:

- улучшение качества сигнала;
- создание постоянного составного канала связи между двумя абонентами сети.

В глобальных сетях необходимо обеспечить качественную передачу сигналов на расстояния в сотни и тысячи километров. Поэтому без усилителей сигналов, установленных через определенные расстояния, построить территориальную линию связи невозможно.

В глобальной сети необходима также и промежуточная аппаратура другого рода — мультиплексоры, демультиплексоры и коммутаторы. Эта аппаратура решает вторую указанную задачу, то есть создает между двумя абонентами сети составной канал из некоммутируемых отрезков физической среды — кабелей с усилителями.

Важно отметить, что приведенные на слайде мультиплексоры, демультиплексоры и коммутаторы образуют составной канал на долговременной основе, например на месяц или год, причем абонент не может влиять на процесс коммутации этого канала — эти устройства управляются по отдельным входам, абоненту недоступным (на рисунке не показаны). Наличие промежуточной коммутационной аппаратуры избавляет создателей глобальной сети от необходимости прокладывать отдельную кабельную линию для каждой пары соединяемых узлов сети. Вместо этого между мультиплексорами и коммутаторами используется высокоскоростная физическая среда, например волоконно-оптический или коаксиальный кабель, по которому передаются одновременно данные от большого числа сравнительно низкоскоростных абонентских линий. Высокоскоростной канал обычно называют **уплотненным каналом.**

Промежуточная аппаратура канала связи прозрачна для пользователя, он ее не замечает и не учитывает в своей работе. Для него важны только качество полученного канала.

В действительности же промежуточная аппаратура образует сложную сеть, которую называют **первичной сетью**, так как сама по себе она никаких высокоуровневых служб (например, файловой или передачи голоса) не поддерживает, а только служит основой для построения компьютерных, телефонных или иных сетей.

В зависимости от типа промежуточной аппаратуры все линии связи делятся на аналоговые и цифровые.

В аналоговых линиях промежуточная аппаратура предназначена для усиления аналоговых сигналов, то есть сигналов, которые имеют непрерывный диапазон значений. Такие линии связи традиционно применялись в телефонных сетях для связи АТС между собой. Для создания высокоскоростных каналов, которые мультиплексируют несколько низкоскоростных аналоговых абонентских каналов, при аналоговом подходе обычно используется техника частотного мультиплексирования (FDM).

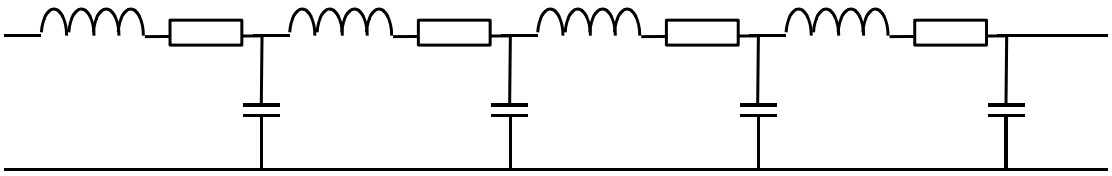
В цифровых линиях связи передаваемые сигналы имеют конечное число состояний. Как правило, элементарный сигнал, то есть сигнал, передаваемый за один такт работы передающей аппаратуры, имеет 2 или 3 состояния, которые передаются в линиях связи импульсами прямоугольной формы

Промежуточная аппаратура образования высокоскоростных цифровых каналов (мультиплексоры, демультиплексоры, коммутаторы) работает по принципу временного мультиплексирования каналов (TDM), когда каждому низкоскоростному каналу выделяется определенная доля времени (**тайм-слот**) высокоскоростного канала.

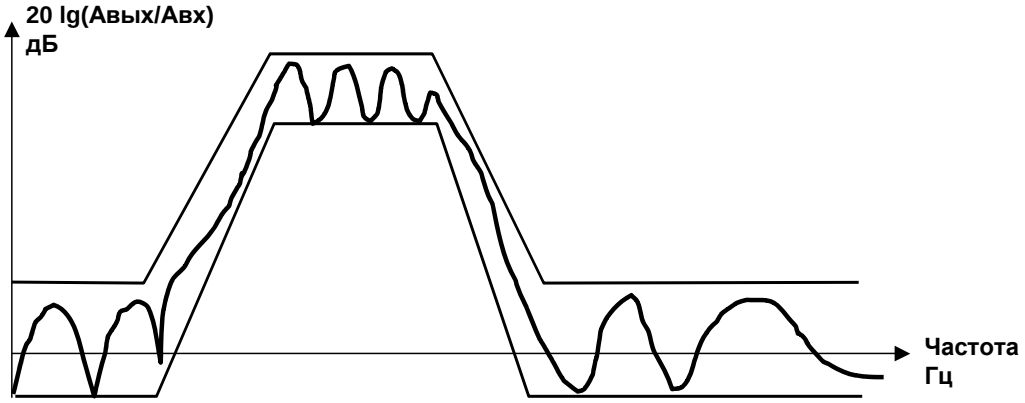
Аппаратура передачи дискретных компьютерных данных по аналоговым и цифровым линиям связи существенно отличается, так как в первом случае линия связи предназначена для передачи сигналов произвольной формы и не предъявляет никаких требований к способу представления единиц и

нулей аппаратурой передачи данных, а во втором — все параметры передаваемых линией импульсов стандартизованы. Другими словами, на цифровых линиях связи протокол физического уровня определен, а на аналоговых линиях — нет.

Характеристики линий связи



Представление линии связи как распределенной индуктивно-емкостной нагрузки



3

Характеристики линий связи.

Линия связи искажает передаваемые данные т.к. ее физические параметры отличаются от идеальных. Линия связи представляет собой некую распределенную комбинацию активного сопротивления, индуктивной и емкостной нагрузки.

Типы характеристик и способы их определения.

К основным характеристикам линий связи относятся:
- амплитудно-частотная характеристика;
- полоса пропускания;
- затухание;
- помехоустойчивость;
- перекрестные наводки на ближнем конце линии;
- пропускная способность;
- достоверность передачи данных;
- удельная стоимость.

В первую очередь разработчика вычислительной сети интересуют пропускная способность и достоверность передачи данных, поскольку эти характеристики прямо влияют на производительность и надежность создаваемой сети. Пропускная способность и достоверность — это характеристики как

линии связи, так и способа передачи данных. Поэтому если способ передачи (протокол) уже определен, то известны и эти характеристики. Например, пропускная способность цифровой линии всегда известна, так как на ней определен протокол физического уровня, который задает битовую скорость передачи данных — 64 Кбит/с, 2 Мбит/с и т. п.

Однако нельзя говорить о пропускной способности линии связи, до того как для нее определен протокол физического уровня.

Амплитудно-частотная характеристика, полоса пропускания и затухание

Амплитудно-частотная характеристика показывает, как затухает амплитуда синусоиды на выходе линии связи по сравнению с амплитудой на ее входе для всех возможных частот передаваемого сигнала. Вместо амплитуды в этой характеристике часто используют также такой параметр сигнала, как его мощность.

На практике вместо АЧХ применяются другие, упрощенные характеристики — полоса пропускания и затухание.

Полоса пропускания — это непрерывный диапазон частот, для которого отношение амплитуды выходного сигнала ко входному превышает некоторый заранее заданный предел, обычно 0,5. Ширина полосы пропускания в наибольшей степени влияет на максимально возможную скорость передачи информации по линии связи.

Затухание определяется как относительное уменьшение амплитуды или мощности сигнала при передаче по линии сигнала определенной частоты. Таким образом, затухание представляет собой одну точку из амплитудно-частотной характеристики линии. Часто при эксплуатации линии заранее известна основная частота передаваемого сигнала, то есть та частота, гармоника которой имеет наибольшую амплитуду и мощность. Поэтому достаточно знать затухание на этой частоте, чтобы приблизительно оценить искажения передаваемых по линии сигналов.

Затухание А обычно измеряется в децибелах и вычисляется по следующей формуле:

$$A = 10 \log (P_{вых}/P_{вх}),$$

Так как мощность выходного сигнала кабеля без промежуточных усилителей всегда меньше, чем мощность входного сигнала, затухание кабеля всегда является отрицательной величиной.

Например, кабель на витой паре категории 5 характеризуется затуханием не ниже -23,6 дБ для частоты 100 МГц при длине кабеля 100 м. Частота 100 МГц выбрана потому, что кабель этой категории предназначен для высокоскоростной передачи данных, сигналы которых имеют значимые гармоники с частотой примерно 100 МГц.

Кабель категории 3 предназначен для низкоскоростной передачи данных, поэтому для него определяется затухание на частоте 10 МГц (не ниже -11,5 дБ). Часто оперируют с абсолютными значениями затухания, без указания знака.

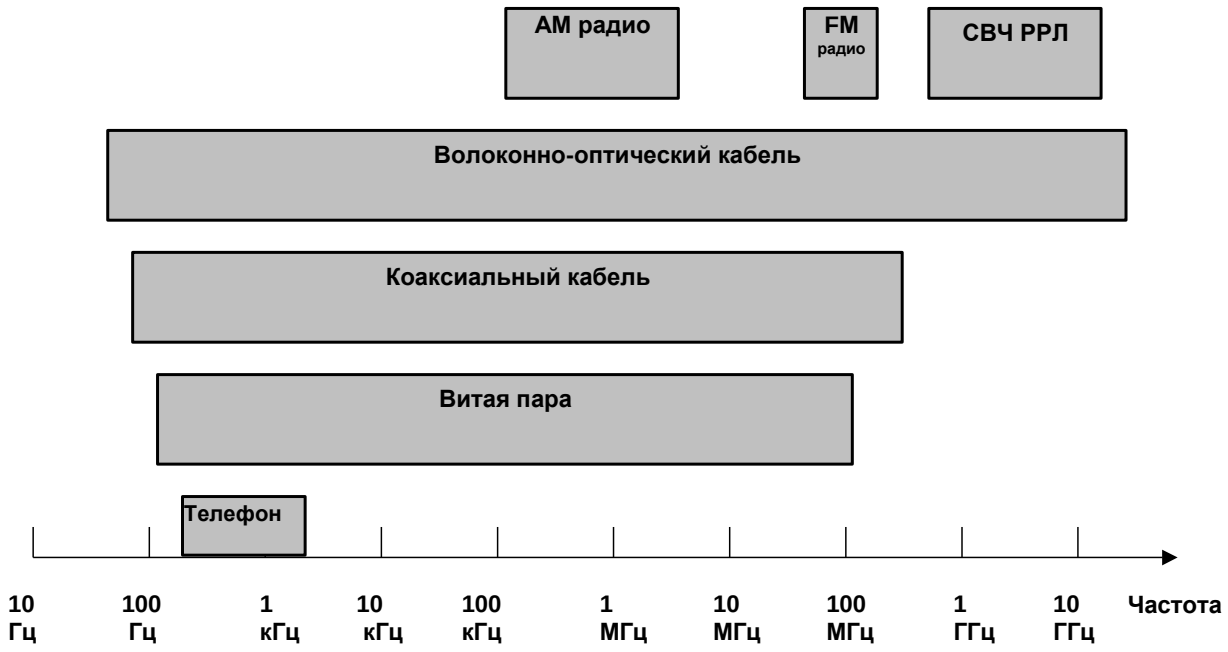
Абсолютный уровень мощности, например уровень мощности передатчика, также измеряется в децибелах. При этом в качестве базового значения мощности сигнала, относительно которого измеряется текущая мощность, принимается значение в 1 мВт. Таким образом, уровень мощности p вычисляется по следующей формуле:

$$p = 10 \log (P/1мВт)\ [дБм],$$

где P — мощность сигнала в милливаттах, а дБм (dBm) — это единица измерения уровня мощности (децибел на 1 мВт).

Таким образом, амплитудно-частотная характеристика, полоса пропускания и затухание являются универсальными характеристиками, и их знание позволяет сделать вывод о том, как через линию связи будут передаваться сигналы любой формы.

Характеристики линий связи



Полосы пропускания линий связи

4

Полоса пропускания зависит от типа линии и ее протяженности. На слайде показаны полосы пропускания линий связи различных типов, а также наиболее часто используемые в технике связи частотные диапазоны.

Пропускная способность линии

Пропускная способность линии характеризует максимально возможную скорость передачи данных по линии связи. Пропускная способность измеряется в битах в секунду — бит/с, а также в производных единицах, таких как килобит в секунду (Кбит/с), мегабит в секунду (Мбит/с), гигабит в секунду (Гбит/с) и т. д.

Выбор способа представления дискретной информации в виде сигналов, подаваемых на линию связи, называется физическим или линейным кодированием. От выбранного способа кодирования зависит спектр сигналов и, соответственно, пропускная способность линии. Таким образом, для одного способа кодирования линия может обладать одной пропускной способностью, а для другого — другой.

Теория информации говорит, что любое различимое и непредсказуемое изменение принимаемого сигнала несет в себе информацию. В соответствии с этим прием синусоиды, у которой амплитуда, фаза и частота остаются неизменными, информации не несет, так как изменение сигнала хотя и происходит, но является хорошо предсказуемым. Аналогично, не несут в себе информации импульсы на тактовой

шине компьютера, так как их изменения также постоянны во времени. А вот импульсы на шине данных предсказать заранее нельзя, поэтому они переносят информацию между отдельными блоками или устройствами.

Если сигнал изменяется так, что можно различить только два его состояния, то любое его изменение будет соответствовать наименьшей единице информации — биту. Если же сигнал может

иметь более двух различимых состояний, то любое его изменение будет нести несколько бит информации.

Количество изменений информационного параметра несущего периодического сигнала в секунду измеряется в бодах.

Пропускная способность линии в битах в секунду в общем случае не совпадает с числом бод. Она может быть как выше, так и ниже числа бод, и это соотношение зависит от способа кодирования.

Если сигнал имеет более двух различимых состояний, то пропускная способность в битах в секунду будет выше, чем число бод.

При использовании сигналов с двумя различимыми состояниями может наблюдаться обратная картина. Это часто происходит потому, что для надежного распознавания приемником пользовательской информации каждый бит в последовательности кодируется с помощью нескольких изменений информационного параметра несущего сигнала. Например, при кодировании единичного значения бита импульсом положительной полярности, а нулевого значения бита — импульсом отрицательной полярности физический сигнал дважды изменяет свое состояние при передаче каждого бита. При таком кодировании пропускная способность линии в два раза ниже, чем число бод, передаваемое по линии.

Например, если информационными параметрами являются фаза и амплитуда синусоиды, причем различаются 4 состояния фазы в 0, 90,180 и 270 градусов и два значения амплитуды сигнала, то информационный сигнал может иметь 8 различимых состояний. В этом случае модем, работающий со скоростью 2400 бод (с тактовой частотой 2400 Гц) передает информацию со скоростью 7200 бит/с, так как при одном изменении сигнала передается 3 бита информации.

На пропускную способность линии оказывает влияние не только физическое, но и логическое кодирование. Логическое кодирование выполняется до физического кодирования и подразумевает замену бит исходной информации новой последовательностью бит, несущей ту же информацию, но обладающей, кроме этого, дополнительными свойствами, например возможностью для приемной стороны обнаруживать ошибки в принятых данных.

Сопровождение каждого байта исходной информации одним битом четности — это пример очень часто применяемого способа логического кодирования при передаче данных с помощью модемов.

Другим примером логического кодирования может служить шифрация данных, обеспечивающая их конфиденциальность при передаче через общественные каналы связи.

При логическом кодировании чаще всего исходная последовательность бит заменяется более длинной последовательностью, поэтому пропускная способность канала по отношению к полезной информации при этом уменьшается.

Характеристики линий связи

$$C=F \log_2(1+Pc/Pш)$$

- формула Шеннона для максимальной пропускной способности

$$C=2F \log_2(M)$$

- соотношение Найквиста

F-ширина полосы пропускания линии связи
M –количество различимых состояний информационного параметра
Pc/Pш – соотношение мощностей сигнала и шума.

$$NEXT= 10 \lg(Pвых/Pнав)$$

– показатель перекрестных наводок на ближнем конце линии.

Связь между пропускной способностью линии и ее полосой пропускания

Чем выше частота несущего периодического сигнала, тем больше информации в единицу времени передается по линии и тем выше пропускная способность линии при фиксированном способе физического кодирования. Однако, с другой стороны, с увеличением частоты периодического несущего сигнала увеличивается и ширина спектра этого сигнала. Линия передает этот спектр синусоид с теми искажениями, которые определяются ее полосой пропускания. **Это не значит, что сигналы нельзя передавать.** Чем больше несоответствие между полосой пропускания линии и шириной спектра передаваемых информационных сигналов, тем больше сигналы искажаются и тем вероятнее ошибки в распознавании информации принимающей стороной, а значит, скорость передачи информации на самом деле оказывается меньше, чем можно было предположить.

Связь между полосой пропускания линии и ее максимально возможной пропускной способностью, вне зависимости от принятого способа физического кодирования, установил Клод Шеннон:

$$C = F \log_2 (1 + Pc/Pш),$$

где C — максимальная пропускная способность линии в битах в секунду,
F — ширина полосы пропускания линии в герцах,
Pc — мощность сигнала,
Pш — мощность шума.

Из этого соотношения видно, что хотя теоретического предела пропускной способности линии с фиксированной полосой пропускания не существует, на практике такой предел имеется. Действительно, повысить пропускную способность линии можно за счет увеличения мощности передатчика или же

уменьшения мощности шума (помех) на линии связи. Обе эти составляющие поддаются изменению с большим трудом. Повышение мощности передатчика ведет к значительному увеличению его габаритов и стоимости. Снижение уровня шума требует применения специальных кабелей с хорошими защитными экранами, что весьма дорого, а также снижения шума в передатчике и промежуточной аппаратуре, чего достичь весьма не просто.

К тому же влияние мощностей полезного сигнала и шума на пропускную способность ограничено логарифмической зависимостью, которая растет далеко не так быстро, как прямо-пропорциональная. Так, при достаточно типичном исходном отношении мощности сигнала к мощности шума в 100 раз повышение мощности передатчика в 2 раза даст только 15 % увеличения пропускной способности линии.

Близким по сути к формуле Шеннона является следующее соотношение, полученное Найквистом, которое также определяет максимально возможную пропускную способность линии связи, но без учета шума на линии:

$$C = 2F \log_2 M,$$

где M — количество различимых состояний информационного параметра.

Если сигнал имеет 2 различимых состояния, то пропускная способность равна удвоенному значению ширины полосы пропускания линии связи. Если же передатчик использует более чем 2 устойчивых состояния сигнала для кодирования данных, то пропускная способность линии повышается, так как за один такт работы передатчик передает несколько бит исходных данных, например 2 бита при наличии четырех различимых состояний сигнала.

Хотя формула Найквиста явно не учитывает наличие шума, косвенно его влияние отражается в выборе количества состояний информационного сигнала. Для повышения пропускной способности канала хотелось бы увеличить это количество до значительных величин, но на практике мы не можем этого сделать из-за шума на линии.

Например можно увеличить пропускную способность линии еще в два раза, использовав для кодирования данных не 4, а 16 уровней. Однако если амплитуда шума часто превышает разницу между соседними 16-ю уровнями, то приемник не сможет устойчиво распознавать передаваемые данные. Поэтому количество возможных состояний сигнала фактически ограничивается соотношением мощности сигнала и шума, а формула Найквиста определяет предельную скорость передачи данных в том случае, когда количество состояний уже выбрано с учетом возможностей устойчивого распознавания приемником.

Приведенные соотношения дают предельное значение пропускной способности линии, а степень приближения к этому пределу зависит от конкретных методов физического кодирования.

Помехоустойчивость и достоверность

Помехоустойчивость линии определяет ее способность уменьшать уровень помех, создаваемых во внешней среде, на внутренних проводниках. Помехоустойчивость линии зависит от типа используемой физической среды, а также от экранирующих и подавляющих помехи средств самой линии. Наименее помехоустойчивыми являются радиолинии, хорошей устойчивостью обладают кабельные линии и отличной — волоконно-оптические линии, малочувствительные ко внешнему электромагнитному излучению. Обычно для уменьшения помех, появляющихся из-за внешних электромагнитных полей, проводники экранируют и/или скручивают.

Перекрестные наводки на ближнем конце (Near End Cross Talk — NEXT) определяют помехоустойчивость кабеля к внутренним источникам помех, когда электромагнитное поле сигнала, передаваемого выходом передатчика по одной паре проводников, наводит на другую пару проводников сигнал помехи. Если ко второй паре будет подключен приемник, то он может принять наведенную внутреннюю помеху за полезный сигнал. Показатель NEXT, выраженный в децибелах, равен

$10 \log (P_{вых}/P_{нав})$, где $P_{вых}$ — мощность выходного сигнала, $P_{нав}$ — мощность наведенного сигнала.

Чем меньше значение NEXT, тем лучше кабель. Так, для витой пары категории 5 показатель NEXT должен быть меньше -27 дБ на частоте 100 МГц.

Показатель NEXT обычно используется применительно к кабелю, состоящему из нескольких витых пар, так как в этом случае взаимные наводки одной пары на другую могут достигать значительных величин. Для одинарного коаксиального кабеля (то есть состоящего из одной экранированной жилы) этот показатель не имеет смысла, а для двойного коаксиального кабеля он также не применяется вследствие высокой степени защищенности каждой жилы. Оптические волокна также не создают сколь-нибудь заметных помех друг для друга.

Достоверность передачи данных характеризует вероятность искажения для каждого передаваемого бита данных. Иногда этот же показатель называют интенсивностью битовых ошибок (Bit Error Rate, BER). Величина BER для каналов связи без дополнительных средств защиты от ошибок (например, самокорректирующихся кодов или протоколов с повторной передачей искаженных кадров) составляет, как правило, 10^{-4}-10^{-6}, в оптоволоконных линиях связи — 10^{-9}. Значение достоверности передачи данных, например, в 10^{-4} говорит о том, что в среднем из 10 000 бит искажается значение одного бита.

Искажения бит происходят как из-за наличия помех на линии, так и по причине искажений формы сигнала ограниченной полосой пропускания линии. Поэтому для повышения достоверности передаваемых данных нужно повышать степень помехозащищенности линии, снижать уровень перекрестных наводок в кабеле, а также использовать более широкополосные линии связи.

Стандарты кабелей

В стандартах кабелей оговаривается достаточно много характеристик, из которых наиболее важные перечислены ниже (первые две из них уже были достаточно детально рассмотрены).

- Затухание (Attenuation). Затухание измеряется в децибелах на метр для определенной частоты или диапазона частот сигнала,
- Перекрестные наводки на ближнем конце (Near End Cross Talk, NEXT). Измеряются в децибелах для определенной частоты сигнала.
- Импеданс (волновое сопротивление) — это полное (активное и реактивное) сопротивление в электрической цепи. Импеданс измеряется в Омах и является относительно постоянной величиной для кабельных систем (например, для коаксиальных кабелей, используемых в стандартах Ethernet, импеданс кабеля должен составлять 50 Ом). Для неэкранированной витой пары наиболее часто используемые значения импеданса — 100 и 120 Ом. В области высоких частот (100-200 МГц) импеданс зависит от частоты.
- Активное сопротивление — это сопротивление постоянному току в электрической цепи. В отличие от импеданса активное сопротивление не зависит от частоты и возрастает с увеличением длины кабеля.
- Емкость — это свойство металлических проводников накапливать энергию. Емкость является нежелательной величиной, поэтому следует стремиться к тому, чтобы она была как можно меньше (иногда применяют термин «паразитная емкость»). Высокое значение емкости в кабеле приводит к искажению сигнала и ограничивает полосу пропускания линии.
- Уровень внешнего электромагнитного излучения или электрический шум. Электрический шум — это нежелательное переменное напряжение в проводнике. Электрический шум бывает двух типов: фоновый и импульсный. Электрический шум можно также разделить на низко-, средне- и высокочастотный. Источниками фонового электрического шума в диапазоне до 150 кГц являются линии электропередачи, телефоны и лампы дневного света; в диапазоне от 150 кГц до 20 МГц — компьютеры, принтеры, ксероксы; в диапазоне от 20 МГц до 1 ГГц — телевизионные и радиопередатчики, микроволновые печи. Основными источниками импульсного электрического шума являются моторы, переключатели и сварочные агрегаты. Электрический шум измеряется в милливольтах.
- Диаметр или площадь сечения проводника. Для медных проводников достаточно употребительной является американская система AWG (American Wire Gauge), которая вводит

некоторые условные типы проводников, например 22 AWG, 24 AWG, 26 AWG. Чем больше номер типа проводника, тем меньше его диаметр.

Помимо универсальных характеристик, таких, например, как затухание, которые применимы для всех типов кабелей, существуют характеристики, которые применимы только к определенному типу кабеля. Например, параметр шаг скрутки проводов используется только для характеристики витой пары, а параметр NEXT применим только к многопарным кабелям на основе витой пары.

Основное внимание в современных стандартах уделяется кабелям на основе витой пары и волоконно-оптическим кабелям.

Волоконно-оптические кабели

Волоконно-оптические кабели состоят из центрального проводника света (сердцевины) — стеклянного волокна, окруженного другим слоем стекла — оболочкой, обладающей меньшим показателем преломления, чем сердцевина. Распространяясь по сердцевине, лучи света не выходят за ее пределы, отражаясь от покрывающего слоя оболочки. В зависимости от распределения показателя преломления и от величины диаметра сердечника различают:

- многомодовое волокно со ступенчатым изменением показателя преломления;
- многомодовое волокно с плавным изменением показателя преломления;
- одномодовое волокно.

Одномодовый кабель (Single Mode Fiber, SMF):

- центральный проводник очень малого диаметра, соизмеримого с длиной волны света - от 5 до 10 мкм. При этом практически все лучи света распространяются вдоль оптической оси световода, не отражаясь от внешнего проводника
- полоса пропускания очень широкая - до сотен гигагерц на километр
- изготовление тонких качественных волокон для одномодового кабеля представляет сложный технологический процесс, что делает одномодовый кабель достаточно дорогим
- в волокно такого маленького диаметра достаточно сложно направить пучок света, не потеряв при этом значительную часть его энергии.

Многомодовый кабель (Multi Mode Fiber, MMF):

- более широкие внутренние сердечники, которые легче изготовить технологически
- в многомодовых кабелях во внутреннем проводнике одновременно существует несколько световых лучей, отражающихся от внешнего проводника под разными углами. Угол отражения луча называется модой луча
- имеют более узкую полосу пропускания — от 500 до 800 МГц/км. Сужение полосы происходит из-за потерь световой энергии при отражениях, а также из-за интерференции лучей разных мод.

В качестве источников излучения света в волоконно-оптических кабелях применяются:

- светодиоды;
- полупроводниковые лазеры.

Для одномодовых кабелей применяются только полупроводниковые лазеры, так как при таком малом диаметре оптического волокна световой поток, создаваемый светодиодом, невозможно без больших потерь направить в волокно. Для многомодовых кабелей используются более дешевые светодиодные излучатели.

Методу передачи дискретных данных на -физическом уровне

1. Модуляция.
2. Цифровое кодирование.

При амплитудной модуляции для логической единицы выбирается один уровень амплитуды синусоиды несущей частоты, а для логического нуля — другой. Этот способ редко используется в чистом виде на практике из-за низкой помехоустойчивости, но часто применяется в сочетании с другим видом модуляции — фазовой модуляцией.

При частотной модуляции значения 0 и 1 исходных данных передаются синусоидами с различной частотой.

При фазовой модуляции значениям данных 0 и 1 соответствуют сигналы одинаковой частоты, но с различной фазой, например 0 и 180 градусов или 0,90,180 и 270 градусов.

В скоростных модемах часто используются комбинированные методы модуляции, как правило, амплитудная в сочетании с фазовой.

Спектр модулированного сигнала

Рассмотрим сначала спектр сигнала при потенциальном кодировании. Пусть логическая единица кодируется положительным потенциалом, а логический ноль — отрицательным потенциалом такой же величины. Для упрощения вычислений предположим, что передается информация, состоящая из бесконечной последовательности чередующихся единиц и нулей, в данном случае величины бод и бит в секунду совпадают.

Если дискретные данные передаются с битовой скоростью N бит/с, то спектр состоит из постоянной составляющей нулевой частоты и бесконечного ряда гармоник с частотами f0,3fo, 5f0,7f0,..., где f0 = N/2. Амплитуды этих гармоник убывают достаточно медленно — с коэффициентами 1/3, 1/5,1/7,... от амплитуды гармоники f0. В результате спектр потенциального кода требует для качественной передачи широкую полосу пропускания. Кроме того, нужно учесть, что реально спектр сигнала постоянно меняется в зависимости от того, какие данные передаются по линии связи. Например, передача длинной последовательности нулей или единиц сдвигает спектр в сторону низких частот, а в крайнем случае, когда передаваемые данные состоят только из единиц (или только из нулей), спектр состоит из гармоники нулевой частоты. При передаче чередующихся единиц и нулей постоянная составляющая отсутствует. Поэтому спектр результирующего сигнала потенциального кода при передаче произвольных данных занимает полосу от некоторой величины, близкой к 0 Гц, до примерно 7fo (гармониками d частотами выше 7fo можно пренебречь из-за их малого вклада в результирующий сигнал).

Для канала тональной частоты (телефон) верхняя граница при потенциальном кодировании достигается для скорости передачи данных в 971 бит/с, а нижняя неприемлема для любых скоростей, так как полоса пропускания канала начинается с 300 Гц. В результате потенциальные коды на каналах тональной частоты никогда не используются.

Цифровое кодирование.

Применяют потенциальные и импульсные коды. Потенциальные для представления сигнала используют уровень, импульсные – импульс либо перепад.

Требования к методам цифрового кодирования:

- Битовая синхронизация приемника и передатчика
- Распознавание ошибочных символов
- Отсутствие постоянной составляющей для обеспечения гальванической развязки
- Экономичное использование частотного спектра

Синхронизация.

- отдельной тактирующей линии связи (на небольших расстояниях)
- самосинхронизирующиеся коды, сигналы которых несут для передатчика указания о том, в какой момент времени нужно осуществлять распознавание очередного бита (или нескольких бит, если код ориентирован более чем на два состояния сигнала). Любой резкий перепад сигнала — так

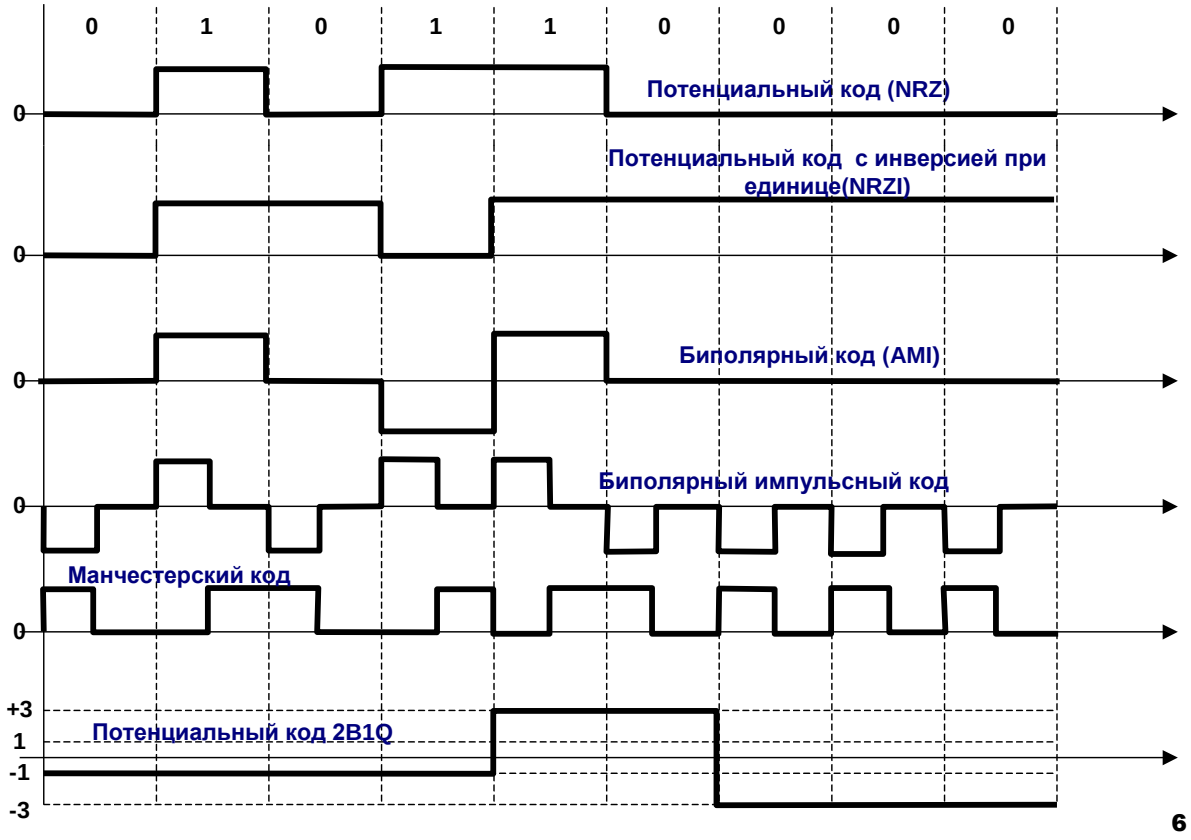
называемый фронт — может служить хорошим указанием для синхронизации приемника с передатчиком.

При использовании синусоид в качестве несущего сигнала результирующий код обладает свойством самосинхронизации, так как изменение амплитуды несущей частоты дает возможность приемнику определить момент появления входного кода.

Распознавание и коррекцию искаженных данных сложно осуществить средствами физического уровня, поэтому чаще всего эту работу берут на себя протоколы, лежащие выше: канальный, сетевой, транспортный или прикладной. С другой стороны, распознавание ошибок на физическом уровне экономит время, так как приемник не ждет полного помещения кадра в буфер, а отбраковывает его сразу при распознавании ошибочных бит внутри кадра.

Требования, предъявляемые к методам кодирования, являются взаимно противоречивыми, поэтому каждый из рассматриваемых ниже популярных методов цифрового кодирования обладает своими преимуществами и своими недостатками по сравнению с другими.

Дискретное кодирование данных



Потенциальный код без возвращения к нулю

Плюсы

- прост в реализации,
- обладает хорошей распознаваемостью ошибок (из-за двух резко отличающихся потенциалов),

Минусы

- не обладает свойством самосинхронизации. При передаче длинной последовательности единиц или нулей сигнал на линии не изменяется, поэтому приемник лишен возможности определять по входному сигналу моменты времени, когда нужно в очередной раз считывать данные. Даже при наличии высокоточного тактового генератора приемник может ошибиться с моментом съема данных, так как

частоты двух генераторов никогда не бывают полностью идентичными. Поэтому при высоких скоростях обмена данными и длинных последовательностях единиц или нулей небольшое рассогласование тактовых частот может привести к ошибке в целый такт и, соответственно, считыванию некорректного значения бита.

- наличие низкочастотной составляющей, которая приближается к нулю при передаче длинных последовательностей единиц или нулей. Из-за этого многие каналы связи, не обеспечивающие прямого гальванического соединения между приемником и источником, этот вид кодирования не поддерживают. В результате в чистом виде код NRZ в сетях не используется. Тем не менее используются его различные модификации, в которых устраняют как плохую самосинхронизацию кода NRZ, так и наличие постоянной составляющей. Привлекательность кода NRZ, из-за которой имеет смысл заняться его улучшением, состоит в достаточно низкой частоте основной гармоники fo, которая равна N/2 Гц, как это было показано в предыдущем разделе. У других методов кодирования, например манчестерского, основная гармоника имеет более высокую частоту.

Метод биполярного кодирования с альтернативной инверсией

Для кодирования логического нуля используется нулевой потенциал, а логическая единица кодируется либо положительным потенциалом, либо отрицательным, при этом потенциал каждой новой единицы противоположен потенциалу предыдущей.

- частично ликвидирует проблемы постоянной составляющей и отсутствия самосинхронизации, присущие коду NRZ. Это происходит при передаче длинных последовательностей единиц. В этих случаях сигнал на линии представляет собой последовательность разнополярных импульсов с тем же спектром, что и у кода NRZ, передающего чередующиеся нули и единицы, то есть без постоянной составляющей и с, основной гармоникой N/2 Гц (где N — битовая скорость передачи данных). Длинные же последовательности нулей также опасны для кода AMI, как и для кода NRZ — сигнал вырождается в постоянный потенциал нулевой амплитуды. Поэтому код AMI требует дальнейшего улучшения, хотя задача упрощается — осталось справиться только с последовательностями нулей.
- для различных комбинаций бит на линии использование кода AMI приводит к более узкому спектру сигнала, чем для кода NRZ, а значит, и к более высокой пропускной способности линии. Например, при передаче чередующихся единиц и нулей основная гармоника fo имеет частоту N/4 Гц. Код AMI предоставляет также некоторые возможности по распознаванию ошибочных сигналов. Так, нарушение строгого чередования полярности сигналов говорит о ложном импульсе или исчезновении с линии корректного импульса.
- в коде AMI используются не два, а три уровня сигнала на линии. Дополнительный уровень требует увеличение мощности передатчика примерно на 3 дБ для обеспечения той же достоверности приема бит на линии, что является общим недостатком кодов с несколькими состояниями сигнала по сравнению с кодами, которые различают только два состояния.

Потенциальный код с инверсией при единице

Существует код, похожий на AMI, но только с двумя уровнями сигнала. При передаче нуля он передает потенциал, который был установлен в предыдущем такте (то есть не меняет его), а при передаче единицы потенциал инвертируется на противоположный. Этот код называется потенциальным кодом с инверсией при единице

Этот код удобен в тех случаях, когда использование третьего уровня сигнала весьма нежелательно, например в оптических кабелях, где устойчиво распознаются два состояния сигнала - свет и темнота. Для улучшения потенциальных кодов, подобных AMI и NRZI, используются два метода. Первый метод основан на добавлении в исходный код избыточных бит, содержащих логические единицы. Очевидно, что в этом случае длинные последовательности нулей прерываются и код становится самосинхронизирующимся для любых передаваемых данных. Исчезает также постоянная

составляющая, а значит, еще более сужается спектр сигнала. Но этот метод снижает полезную пропускную способность линии, так как избыточные единицы пользовательской информации не несут.

Другой метод основан на предварительном «перемешивании» исходной информации таким образом, чтобы вероятность появления единиц и нулей на линии становилась близкой. Устройства, или блоки, выполняющие такую операцию, называются скрэмблерами (scramble — свалка, беспорядочная сборка). При скремблировании используется известный алгоритм, поэтому приемник, получив двоичные данные, передает их на дескрэмблер, который восстанавливает исходную последовательность бит.

Избыточные биты при этом по линии не передаются. Оба метода относятся к логическому, а не физическому кодированию, так как форму сигналов на линии они не определяют.

Биполярный импульсный код

- обладает отличными самосинхронизирующими свойствами
- но постоянная составляющая может присутствовать, например, при передаче длинной последовательности единиц или нулей
- кроме того, спектр у него шире, чем у потенциальных кодов. Так, при передаче всех нулей или единиц частота основной гармоники кода будет равна N Гц, что в два раза выше основной гармоники кода NRZ и в четыре раза выше основной гармоники кода AMI при передаче чередующихся единиц и нулей. Из-за слишком широкого спектра биполярный импульсный код используется редко.

Манчестерский код

В локальных сетях до недавнего времени самым распространенным методом кодирования был так называемый манчестерский код. Он применяется в технологиях Ethernet и Token Ring.

- для кодирования единиц и нулей используется перепад потенциала, то есть фронт импульса
- каждый такт делится на две части, а информация кодируется перепадами потенциала, происходящими в середине каждого такта. В начале каждого такта может происходить служебный перепад сигнала, если нужно представить несколько единиц или нулей подряд
- так как сигнал изменяется по крайней мере один раз за такт передачи одного бита данных, то манчестерский код обладает хорошими самосинхронизирующими свойствами
- полоса пропускания манчестерского кода уже, чем у биполярного импульсного
- у него также нет постоянной составляющей, а основная гармоника в худшем случае (при передаче последовательности единиц или нулей) имеет частоту N Гц, а в лучшем (при передаче чередующихся единиц и нулей) она равна N/2 Гц, как и у кодов AMI или NRZ.
- в среднем ширина полосы манчестерского кода в полтора раза уже, чем у биполярного импульсного кода, а основная гармоника колеблется вблизи значения 3N/4.
- Манчестерский код имеет еще одно преимущество перед биполярным импульсным кодом. В последнем для передачи данных используются три уровня сигнала, а в манчестерском — два.

Потенциальный код 2B1Q

На рис. 2.16, д показан потенциальный код с четырьмя уровнями сигнала для кодирования данных. Это код 2B1Q название которого отражает его суть — каждые два бита (2B) передаются за один такт сигналом, имеющим четыре состояния (1Q). Паре бит 00 соответствует потенциал -2,5 В, паре бит 01 соответствует потенциал -0,833 В, паре 11 — потенциал +0,833 В, а паре 10 — потенциал +2,5 В.

При этом способе кодирования требуются дополнительные меры по борьбе с длинными последовательностями одинаковых пар бит, так как при этом сигнал превращается в постоянную составляющую. При случайном чередовании бит спектр сигнала в два раза уже, чем у кода NRZ, так как при той же битовой скорости длительность такта увеличивается в два раза.

Таким образом, с помощью кода 2B1Q можно по одной и той же линии передавать данные в два раза быстрее, чем с помощью кода AMI или NRZI.

Однако для его реализации мощность передатчика должна быть выше, чтобы четыре уровня четко различались приемником на фоне помех.

Логическое кодирование

Логическое кодирование используется для улучшения потенциальных кодов типа AMI, NRZI или 2Q1B.

Логическое кодирование должно заменять длинные последовательности бит, приводящие к постоянному потенциалу, вкраплениями единиц.

Для логического кодирования характерны два метода — избыточные коды и скрэмблирование.

Избыточные коды

Основаны на разбиении исходной последовательности бит на порции, которые часто называют символами. Затем каждый исходный символ заменяется на новый, который имеет большее количество бит, чем исходный.

Логическое кодирование

Исходный код	Результирующий код	Исходный код	Результирующий код
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

7

Например, логический код 4B/5B, используемый в технологиях FDDI и Fast Ethernet, меняет исходные символы длиной в 4 бита на символы длиной в 5 бит. Так как результирующие символы содержат избыточные биты, то общее количество битовых комбинаций в них больше, чем в исходных. Так, в коде 4B/5B результирующие символы могут содержать 32 битовых комбинации, в то время как исходные символы — только 16. Поэтому в результирующем коде можно отобрать 16 таких комбинаций, которые не содержат большого количества нулей, а остальные считать запрещенными кодами (code violation).

Кроме устранения постоянной составляющей и придания коду свойства самосинхронизации, избыточные коды позволяют приемнику распознавать искаженные биты. Если приемник принимает запрещенный код, значит, на линии произошло искажение сигнала.

Код 4B/5B затем передается по линии с помощью физического кодирования по одному из методов потенциального кодирования, чувствительному только к длинным последовательностям нулей.

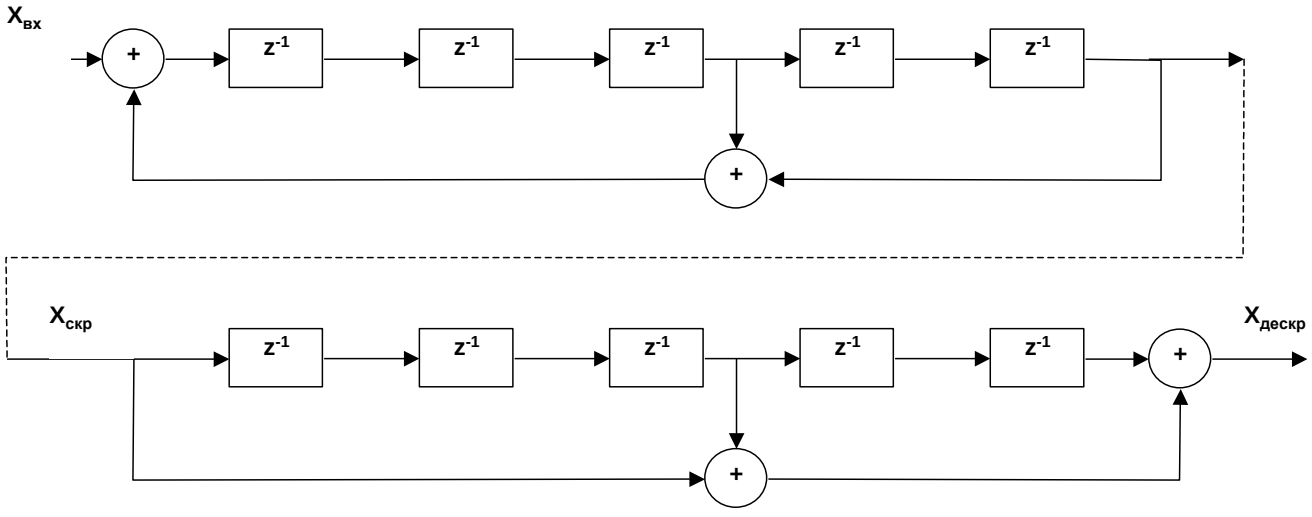
Символы кода 4B/5B длиной 5 бит гарантируют, что при любом их сочетании на линии не могут встретиться более трех нулей подряд.

Буква B в названии кода означает, что элементарный сигнал имеет 2 состояния — от английского binary — двоичный. Имеются также коды и с тремя состояниями сигнала, например, в коде 8B/6T для кодирования 8 бит исходной информации используется код из 6 сигналов, каждый из которых имеет три состояния. Избыточность кода 8B/6T выше, чем кода 4B/5B, так как на 256 исходных кодов приходится 3^6=729 результирующих символов.

Использование таблицы перекодировки является очень простой операцией, поэтому этот подход не усложняет сетевые адаптеры и интерфейсные блоки коммутаторов и маршрутизаторов.

Для обеспечения заданной пропускной способности линии передатчик, использующий избыточный код, должен работать с повышенной тактовой частотой. Так, для передачи кодов 4B/5B со скоростью 100 Мб/с передатчик должен работать с тактовой частотой 125 МГц. При этом спектр сигнала на линии расширяется по сравнению со случаем, когда по линии передается чистый, не избыточный код. Тем не менее спектр избыточного потенциального кода оказывается уже спектра манчестерского кода, что оправдывает дополнительный этап логического кодирования, а также работу приемника и передатчика на повышенной тактовой частоте.

Дискретное кодирование данных



Скремблер и дескремблер информационной последовательности

8

Скрэмблирование

Перемешивание данных скрэмблером перед передачей их в линию с помощью потенциального кода является другим способом логического кодирования.

Методы скрэмблирования заключаются в побитном вычислении результирующего кода на основании бит исходного кода и полученных в предыдущих тактах бит результирующего кода. Например, скрэмблер может реализовывать следующее соотношение:

$B_i = A_i + B_{i-3} + B_{i-5}$,

где Bi — двоичная цифра результирующего кода, полученная на i-м такте работы скрэмблера, Ai - двоичная цифра исходного кода, поступающая на i-м такте на вход скрэмблера, Bi-з и Bi-5 — двоичные цифры результирующего кода, полученные на предыдущих тактах работы скрэмблера, соответственно на 3 и на 5 тактов ране» текущего такта, + операция исключающего ИЛИ (сложение по модулю 2).

Например, для исходной последовательности 110110000001 скрэмблер даст с дующий результирующий код:

B1 = A2 = 1 (первые три цифры результирующего кода будут совпадать с исходным, так как еще нет нужных предыдущих цифр)
B2 = A2 = 1
B3 = A3 = 0
B4= A4 + B1 = 1 + 1 = 0
B5 = A5 + B2 = 1 + 1 = 0
B6 = A6 + B3 + B1 = 0 + 0 + 1 = 1
B7 = A7 + B4 + B2 = 0 + 0 + 1 = 1
B8 = A8 + B5 + B3 = 0 + 0 + 0 = 0
B9 = A9 + B6 + B4 = 0 + 1 + 0 = 1
B10 = A10 + B7 + B5 = 0 + 1 + 0 = 1
B11 = A11 + B8 + B6 = 0 + 0 + 1 = 1
Bl2 = A12 + B9 + B7 = 1 + 1 + 1 = 1

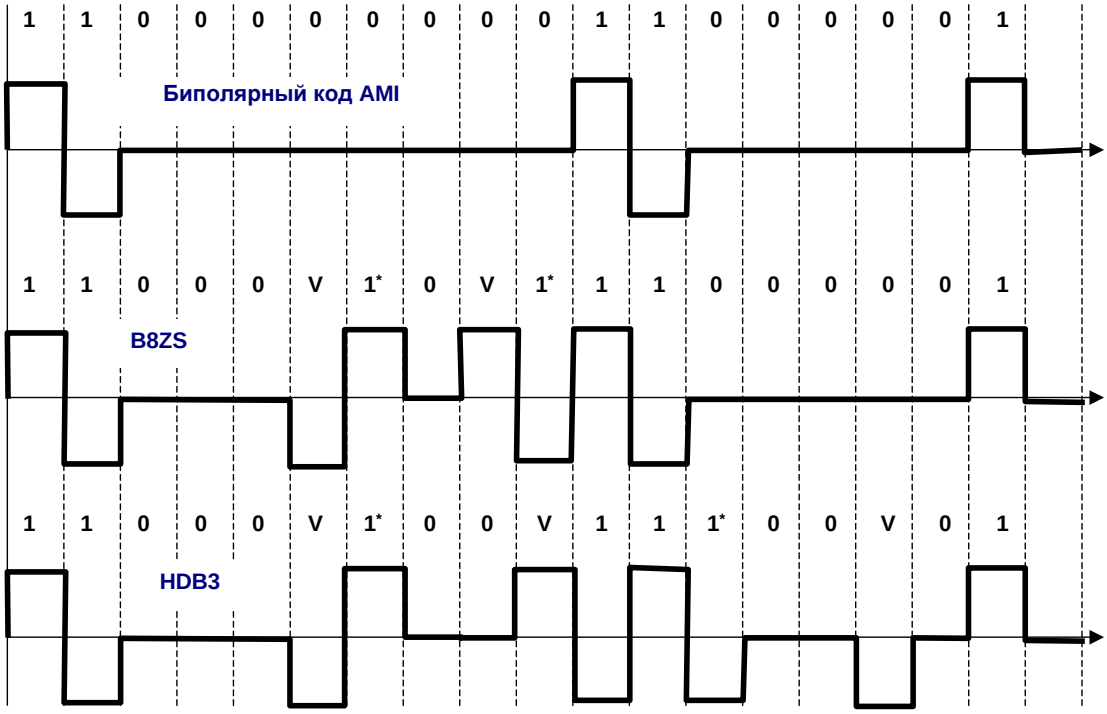
Таким образом, на выходе скрэмблера появится последовательность 110001101111, в которой нет последовательности из шести нулей, присутствовавшей в исходном коде.

После получения результирующей последовательности приемник передает ее дескрэмблеру, который восстанавливает исходную последовательность на основании обратного соотношения:

Ci = Bi + Bi-3 + Bi-5 = (Ai + Bi-3 + Bi-5) + Bi-3 + Bi-5 = A

Различные алгоритмы скрэмблирования отличаются количеством слагаемых, дающих цифру результирующего кода, и сдвигом между слагаемыми. Так, в сетях ISDN при передаче данных от сети к абоненту используется преобразование со сдвигами в 5 и 23 позиции, а при передаче данных от абонента в сеть — со сдвигами 18 и 23 позиции.

Дискретное кодирование данных



Линейные коды высокой плотности B8ZS и HDB3 9

Существуют и более простые методы борьбы с последовательностями единиц, также относимые к классу скрэмблирования.

Для улучшения кода Bipolar AMI используются два метода, основанные на искусственном искажении последовательности нулей запрещенными символами.

B8ZS (Bipolar with 8 Zeros Substitution) и метод HDB3 (High-Density Bipolar 3-Zeros) для корректировки кода AMI.

Исходный код состоит из двух длинных последовательностей нулей: в первом случае — из 8, а во втором — из 5.

Код B8ZS исправляет только последовательности, состоящие из 8 нулей. Для этого он после первых трех нулей вместо оставшихся пяти нулей вставляет пять цифр: V - l* - 0 –V - l*. V здесь обозначает сигнал единицы, запрещенной для данного такта полярности, то есть сигнал, не изменяющий полярность предыдущей единицы, 1* — сигнал единицы корректной полярности, а знак звездочки отмечает тот факт, что в исходном коде в этом такте была не единица, а ноль. В результате на 8 тактах приемник наблюдает 2 искажения — очень маловероятно, что это случилось из-за шума на линии или других сбоев передачи. Поэтому приемник считает такие нарушения кодировкой 8 последовательных нулей и после приема заменяет их на исходные 8 нулей.

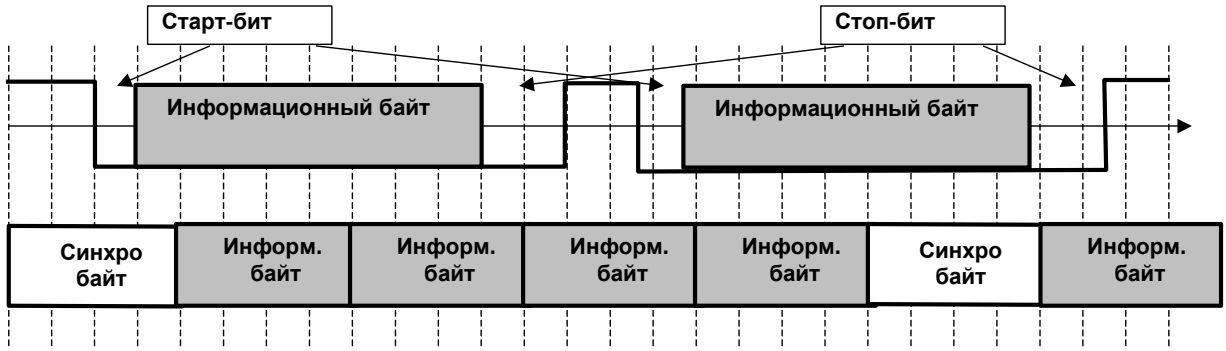
Код B8ZS построен так, что его постоянная составляющая равна нулю при любых последовательностях двоичных цифр.

Код HDB3 исправляет любые четыре подряд идущих нуля в исходной последовательности.

Правила формирования кода HDB3 более сложные, чем кода B8ZS. Каждые четыре нуля заменяются четырьмя сигналами, в которых имеется один сигнал V. Для подавления постоянной составляющей полярность сигнала V чередуется при последовательных заменах. Кроме того, для замены используются два образца четырехтактовых кодов. Если перед заменой исходный код содержал нечетное число единиц, то используется последовательность 000V, а если число единиц было четным — последовательность l*00V.

Улучшенные потенциальные коды обладают достаточно узкой полосой пропускания для любых последовательностей единиц и нулей, которые встречаются в передаваемых данных. Коды, полученные из потенциального путем логического кодирования, обладают более узким спектром, чем манчестерский, даже при повышенной тактовой частоте. Этим объясняется применение потенциальных избыточных и скрэмблированных кодов в современных технологиях, подобных FDDI, Fast Ethernet, Gigabit Ethernet, ISDN и т. п. вместо манчестерского и биполярного импульсного кодирования.

Дискретное кодирование данных



Асинхронная и синхронная передача на уровне байт

Кадровая структура синхронных протоколов

10

Асинхронная и синхронная передачи

В асинхронном режиме каждый байт данных сопровождается специальными сигналами «старт» и «стоп». Назначение этих сигналов состоит в том, чтобы, во-первых, известить приемник о приходе данных и, во-вторых, чтобы дать приемнику достаточно времени для выполнения некоторых функций, связанных с синхронизацией, до поступления следующего байта. Сигнал «старт» имеет продол-жительность в один тактовый интервал, а сигнал «стоп» может длиться один, полтора или два такта, поэтому говорят, что используется один, полтора или два бита в качестве стопового сигнала, хотя пользовательские биты эти сигналы не представляют.

Асинхронным описанный режим называется потому, что каждый байт может быть несколько смещен во времени относительно побитовых тактов предыдущего байта.

При синхронном режиме передачи старт-стопные биты между каждой парой байт отсутствуют. Пользовательские данные собираются в кадр, который предваряется байтами синхронизации. Байт синхронизации — это байт, содержащий заранее известный код, который оповещает приемник о приходе кадра данных. При его получении приемник должен войти в байтовый синхронизм с передатчиком, то есть правильно понимать начало очередного байта кадра. Иногда применяется несколько синхробайт для обеспечения более надежной синхронизации приемника и передатчика.

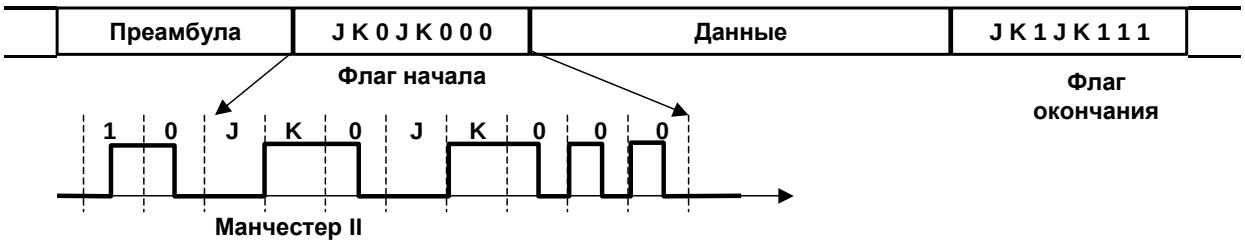
Лекция 5. Основы передачи дискретных данных.

Методы передачи данных канального уровня

Дискретное кодирование данных

Выделение границ кадра с помощью бит - стаффинга

Формат кадра с указанием длины поля данных



Выделение границ кадра с запрещенными символами линейного кода

2

На слайде показаны различные схемы бит-ориентированной передачи. Они отличаются способом обозначения начала и конца каждого кадра.

Первая схема. Начало и конец каждого кадра отмечается одной и той же 8-битовой последовательностью — 01111110, называемой флагом. Термин «бит-ориентированный» используется потому, что принимаемый поток бит сканируется на побитовой основе для обнаружения стартового флага, а затем во время приема для обнаружения стопового флага. Поэтому длина кадра в этом случае не обязательно кратна 8 бит.

Чтобы обеспечить синхронизацию приемника, передатчик посылает последовательность байтов простоя (каждый состоит из 11111111), предшествующую стартовому флагу.

Для достижения прозрачности данных в этой схеме необходимо, чтобы флаг не присутствовал в поле данных кадра. Это достигается с помощью приема, известного как вставка 0 бита, — бит-стаффинга. Схема вставки бита работает на передающей стороне во время передачи поля данных кадра. Если эта схема обнаруживает, что подряд передано пять 1, то она автоматически вставляет дополнительный 0 (даже если после этих пяти 1 шел 0). Поэтому последовательность 01111110 никогда не появится в поле данных кадра. Аналогичная схема работает в приемнике и выполняет обратную функцию. Когда после пяти 1 обнаруживается 0, он автоматически удаляется из поля данных.

Во второй схеме для обозначения начала кадра имеется только стартовый флаг, а для определения конца кадра используется поле длины кадра, которое при фиксированных размерах заголовка и концевика чаще всего имеет смысл длины поля данных. Эта схема наиболее применима в локальных сетях. В этих сетях для обозначения факта незанятости среды в исходном состоянии по

среде вообще не передается никаких символов. Чтобы все остальные станции вошли в битовую синхронизацию, посылающая станция предваряет, содержимое кадра последовательностью бит, известной как преамбула, которая состоит из чередования единиц и нулей 101010... Войдя в битовую синхронизацию, приемник исследует входной поток на побитовой основе, пока не обнаружит байт начала кадра 10101011. За этим байтом следует заголовок кадра, в котором в определенном месте находится поле длины поля данных. Таким образом, в этой схеме приемник просто отсчитывает заданное количество байт, чтобы определить окончание кадра.

Третья схема (внизу) использует для обозначения начала и конца кадра флаги, которые включают запрещенные для данного кода сигналы (V). Например, при манчестерском кодировании вместо обязательного изменения полярности сигнала в середине тактового интервала уровень сигнала остается неизменным и низким (запрещенный сигнал J) или неизменным и высоким (запрещенный сигнал К). Начало кадра отмечается последовательностью JK0JK000, а конец — последовательностью JK1JK100. Этот способ очень экономичен, так как не требует ни бит-стаффинга, ни поля длины, но его недостаток заключается в зависимости от принятого метода физического кодирования. При использовании избыточных кодов (например 4B/5B) роль сигналов J и К играют запрещенные символы.

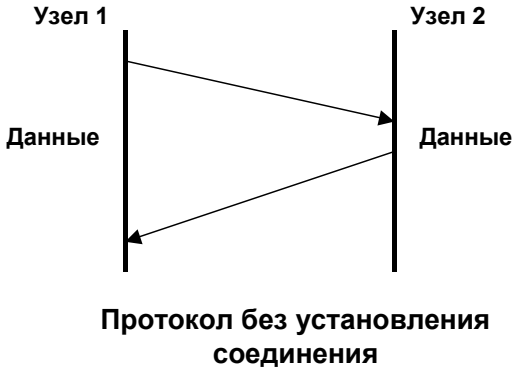
Каждая из трех схем имеет свои преимущества и недостатки. Флаги позволяют отказаться от специального дополнительного поля, но требуют специальных мер: либо бит-стаффинга, либо запрещенных сигналов, что делает эту схему зависимой от способа кодирования.

Для большей части протоколов характерны кадры, состоящие из служебных полей фиксированной длины. Исключение делается только для поля данных, с целью экономной пересылки как небольших квитанций, так и больших файлов. Способ определения окончания кадра путем задания длины поля данных, рассмотренный выше, как раз рассчитан на такие кадры с фиксированной структурой и фиксированными размерами служебных полей.

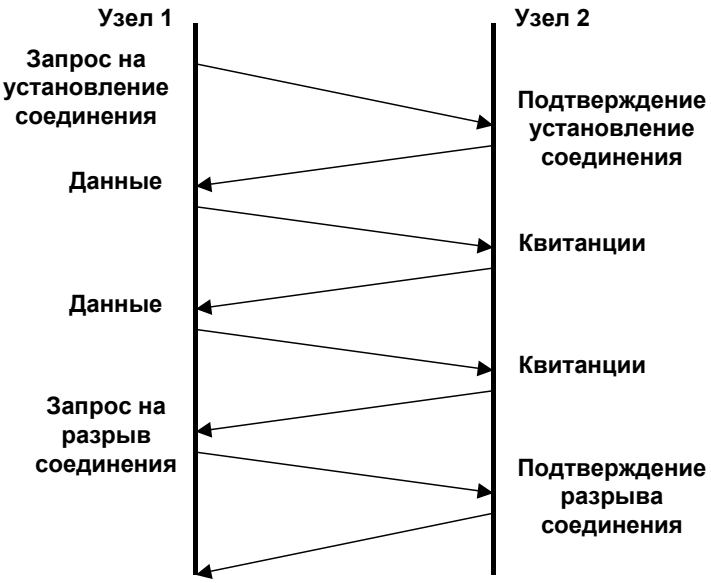
Однако существует ряд протоколов, в которых кадры имеют гибкую структуру (например, PPP). Кадры таких протоколов состоят из неопределенного количества полей, каждое из которых может иметь переменную длину. Начало такого кадра отмечается некоторым стандартным образом, например с помощью флага, а затем протокол последовательно просматривает поля кадра и определяет их количество и размеры. Каждое поле обычно описывается двумя дополнительными полями фиксированного размера.

Передача с установлением соединения и без установления соединения.

Дискретное кодирование данных



Протокол без установления соединения



Протокол с установлением соединения

3

При передаче кадров данных на канальном уровне используются как дейтаграмные процедуры, работающие без установления соединения, так и процедуры с предварительным установлением логического соединения.

При дейтаграммной передаче кадр посылается в сеть «без предупреждения», и никакой ответственности за его утерю протокол не несет (рис. слева). Предполагается, что сеть всегда готова принять кадр.

+ работает быстро, так как никаких предварительных действий перед отправкой данных не выполняется;

- трудно организовать в рамках протокола отслеживание факта доставки (нет гарантии).

Передача с установлением соединения более надежна, но требует больше времени для передачи данных и вычислительных затрат от конечных узлов.

В этом случае узлу-получателю отправляется служебный кадр специального формата с предложением установить соединение (рис. справа). Если узел-получатель согласен с этим, то он посылает в ответ другой служебный кадр, подверждающий установление соединения и предлагающий для данного логиче-кого соединения некоторые параметры, например идентификатор соединения, максимальное значение поля данных кадров, которые будут использоваться в рамках данного соединения, и т. п. Узел-инициатор соединения может завершить процесс установления соединения отправкой третьего служебного кадра, в котором сообщит, что предложенные параметры ему подходят. На этом логическое соединение считается установленным, и в его рамках можно передавать информационные кадры с пользовательскими данными.

После передачи некоторого законченного набора данных, например определенного файла, узел инициирует разрыв данного логического соединения, посылая соответствующий служебный кадр.

В отличие от протоколов дейтаграммного типа, которые поддерживают только один тип кадра — информационный, протоколы, работающие по процедуре с установлением соединения, должны поддерживать несколько типов кадров — служебные, для установления (и разрыва) соединения, и информационные, переносящие собственно пользовательские данные.

Логическое соединение обеспечивает передачу данных как в одном направлении — от инициатора соединения, так и в обоих направлениях.

Процедура установления соединения может использоваться для достижения различных целей.

- Для взаимной аутентификации либо пользователей, либо оборудования.
- Для согласования изменяемых параметров протокола: окна, различные тайм-ауты и т. п.
- Для обнаружения и коррекции ошибок. Установление логического соединения дает точку отсчета для задания начальных значений номеров кадров. При потере нумерованного кадра приемник, во-первых, получает возможность обнаружить этот факт, а во-вторых, он может сообщить передатчику, какой в точности кадр нужно передать повторно.

Далее рассмотрим использование логического соединения для обнаружения и коррекции ошибок. 2.3.4.

Обнаружение и коррекция ошибок.

Большая часть протоколов канального уровня выполняет только одну задачу — обнаружение ошибок, считая, что корректировать ошибки, то есть повторно передавать данные, должны протоколы верхних уровней. Однако существуют протоколы канального уровня, которые самостоятельно решают задачу восстановления искаженных или потерянных кадров.

Но нельзя считать, что один протокол лучше другого потому, что он восстанавливает ошибочные кадры, а другой протокол — нет. Каждый протокол должен работать в тех условиях, для которых он разработан.

Методы обнаружения ошибок

Все методы обнаружения ошибок основаны на передаче служебной избыточной информации, по которой можно судить с некоторой степенью вероятности о достоверности принятых данных. Эту служебную информацию принято называть контрольной суммой. Контрольная сумма вычисляется как функция от основной информации, причем необязательно только путем суммирования. Принимающая сторона повторно вычисляет контрольную сумму кадра по известному алгоритму и в случае ее совпадения делает вывод о том, что данные были переданы корректно. Существует несколько распространенных алгоритмов вычисления контрольной суммы, отличающихся вычислительной сложностью и способностью обнаруживать ошибки в данных.

Контроль по паритету – наиболее простой и наименее мощный метод контроля. С его помощью можно обнаружить только одиночные ошибки в проверяемых данных. Метод заключается в суммировании по модулю 2 всех бит контролируемой информации. Результат суммирования также представляет собой один бит данных. При искажении любого одного бита исходных данных (или контрольного разряда) результат суммирования будет отличаться от принятого контрольного разряда, что говорит об ошибке. Однако двойная ошибка будет неверно принята за корректные данные. Поэтому контроль по паритету применяется к небольшим порциям данных, как правило, к каждому байту, что дает коэффициент избыточности для этого метода 1/8. Метод редко применяется в вычислительных сетях из-за его большой избыточности и невысоких диагностических способностей.

Вертикальный и горизонтальный контроль по паритету представляет собой модификацию описанного выше метода. Его отличие состоит в том, что исходные данные рассматриваются в виде

матрицы, строки которой составляют байты данных. Контрольный разряд подсчитывается отдельно для каждой строки и для каждого столбца матрицы. Этот метод обнаруживает большую часть двойных ошибок, однако обладает еще большей избыточностью. На практике сейчас также почти не применяется.

Циклический избыточный контроль (CRC) является в настоящее время наиболее популярным методом контроля в вычислительных сетях (и не только в сетях, например, этот метод широко применяется при записи данных на диски и дискеты). Метод основан на рассмотрении исходных данных в виде одного многоразрядного двоичного числа. Например, кадр стандарта Ethernet, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 бит. В качестве контрольной информации рассматривается остаток от деления этого числа на известный делитель R. Обычно в качестве делителя выбирается семнадцати- или тридцати трехразрядное число, чтобы остаток от деления имел длину 16 разрядов (2 байт) или 32 разряда (4 байт). При получении кадра данных снова вычисляется остаток от деления на тот же делитель R, но при этом к данным кадра добавляется и содержащаяся в нем контрольная сумма. Если остаток от деления на R равен нулю1, то делается вывод об отсутствии ошибок в полученном кадре, в противном случае кадр считается искаженным.

Этот метод обладает более высокой вычислительной сложностью, но его диагностические возможности гораздо выше, чем у методов контроля, по паритету. Метод CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе бит. Метод обладает также невысокой степенью избыточности. Например, для кадра Ethernet размером в 1024 байт контрольная информация длиной в 4 байт составляет только 0,4 %.

Методы восстановления искаженных и потерянных кадров.

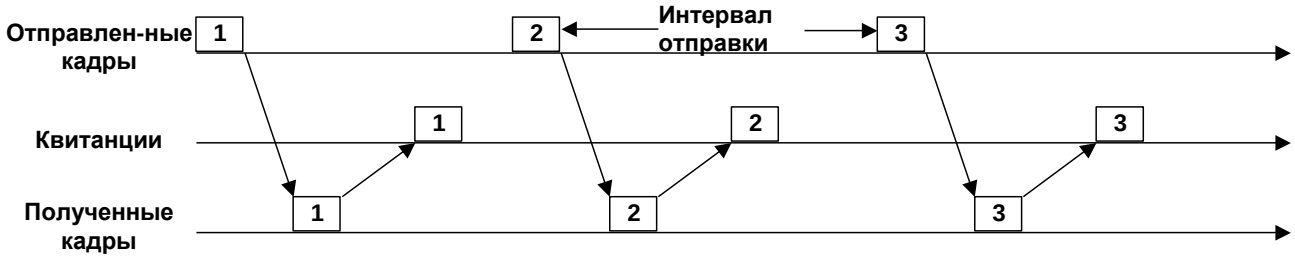
Методы коррекции ошибок в вычислительных сетях основаны на повторной передаче кадра данных.

Чтобы убедиться в необходимости повторной передачи данных,

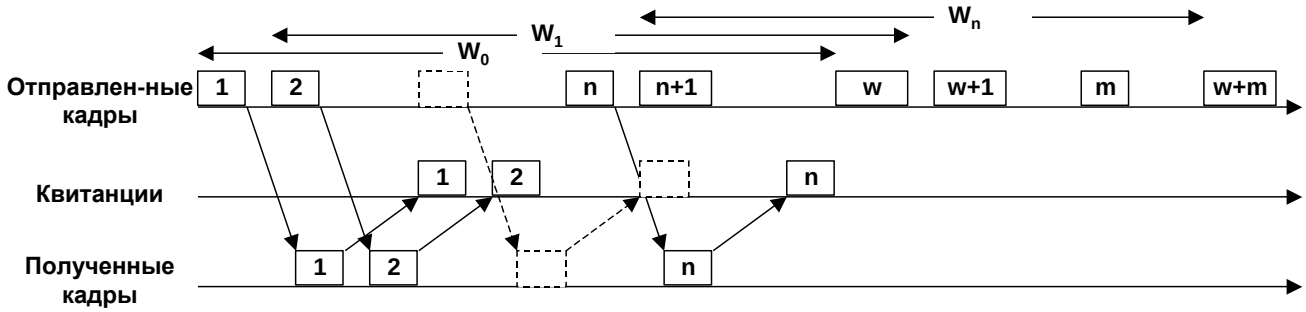
- отправитель нумерует отправляемые кадры и для каждого кадра ожидает от приемника квитанции — служебного кадра, извещающего о том, что исходный кадр был получен и данные в нем оказались корректными.
- время этого ожидания ограничено — при отправке каждого кадра передатчик запускает таймер, и, если по его истечении положительная квитанция на получена, кадр считается утерянным.
- приемник в случае получения кадра с искаженными данными может отправить отрицательную квитанцию — явное указание на то, что данный кадр нужно передать повторно.

Существуют два подхода к организации процесса обмена квитанциями: с простоями и с организацией «окна».

Обнаружение и коррекция ошибок



Метод с простоями



Метод скользящего окна

4

Метод с простоями требует, чтобы источник, пославший кадр, ожидал получения квитанции (положительной или отрицательной) от приемника и только после этого посылал следующий кадр (или повторял искаженный). Если же квитанция не приходит в течение тайм-аута, то кадр (или квитанция) считается утерянным и его передача повторяется. На рис. сверху видно, что в этом случае производительность обмена данными существенно снижается, — хотя передатчик и мог бы послать следующий кадр сразу же после отправки предыдущего, он обязан ждать прихода квитанции.

Второй метод называется методом **«скользящего окна»**. В этом методе для повышения коэффициента использования линии источнику разрешается передать некоторое количество кадров в непрерывном режиме, то есть в максимально возможном для источника темпе, без получения на эти кадры положительных ответных квитанций. Количество кадров, которые разрешается передавать таким образом, называется размером окна. Рисунок внизу иллюстрирует данный метод для окна размером в W кадров.

В начальный момент, когда еще не послано ни одного кадра, окно определяет диапазон кадров с номерами от 1 до W включительно. Источник начинает передавать кадры и получать в ответ квитанции. Для простоты предположим, что квитанции поступают в той же последовательности, что и кадры, которым они соответствуют. При получении первой квитанции окно сдвигается на одну позицию, определяя новый диапазон от 2 до (W+1).

Процессы отправки кадров и получения квитанций идут достаточно независимо друг от друга. Рассмотрим произвольный момент времени tn, когда источник получил квитанцию на кадр с номером n. Окно сдвинулось вправо и определило диапазон разрешенных к передаче кадров от (n+1) до (W+n). Все множество кадров, выходящих из источника, можно разделить на несколько групп:

- Кадры с номерами от1 до n уже были отправлены и квитанции на них получены, то есть они находятся за пределами окна слева.
- Кадры, начиная с номера (n+1) и кончая номером (W+n), находятся в пределах окна и потому могут быть отправлены не дожидаясь прихода какой-либо квитанции. Этот диапазон может быть разделен еще на два поддиапазона:
  - кадры с номерами от (n+1) до m которые уже отправлены, но квитанции на них еще не получены;
  - кадры с номерами от m до (W+n), которые пока не отправлены, хотя запрета на это нет.
- Все кадры с номерами, большими или равными (W+n+1), находятся за пределами окна справа и поэтому пока не могут быть отправлены.

Хотя в данном примере размер окна в процессе передачи остается постоянным, в реальных протоколах (например, TCP) можно встретить варианты данного алгоритма с изменяющимся размером окна.

Итак, при отправке кадра с номером n источнику разрешается передать еще W-1 кадров до получения квитанции на кадр n, так что в сеть последним уйдет кадр с номером (W+n-1). Если же за это время квитанция на кадр n так и не пришла, то процесс передачи приостанавливается, и по истечении некоторого тайм-аута кадр n (или квитанция на него) считается утерянным, и он передается снова.

Если же поток квитанций поступает более-менее регулярно, в пределах допуска в W кадров, то скорость обмена достигает максимально возможной величины для данного канала и принятого протокола.

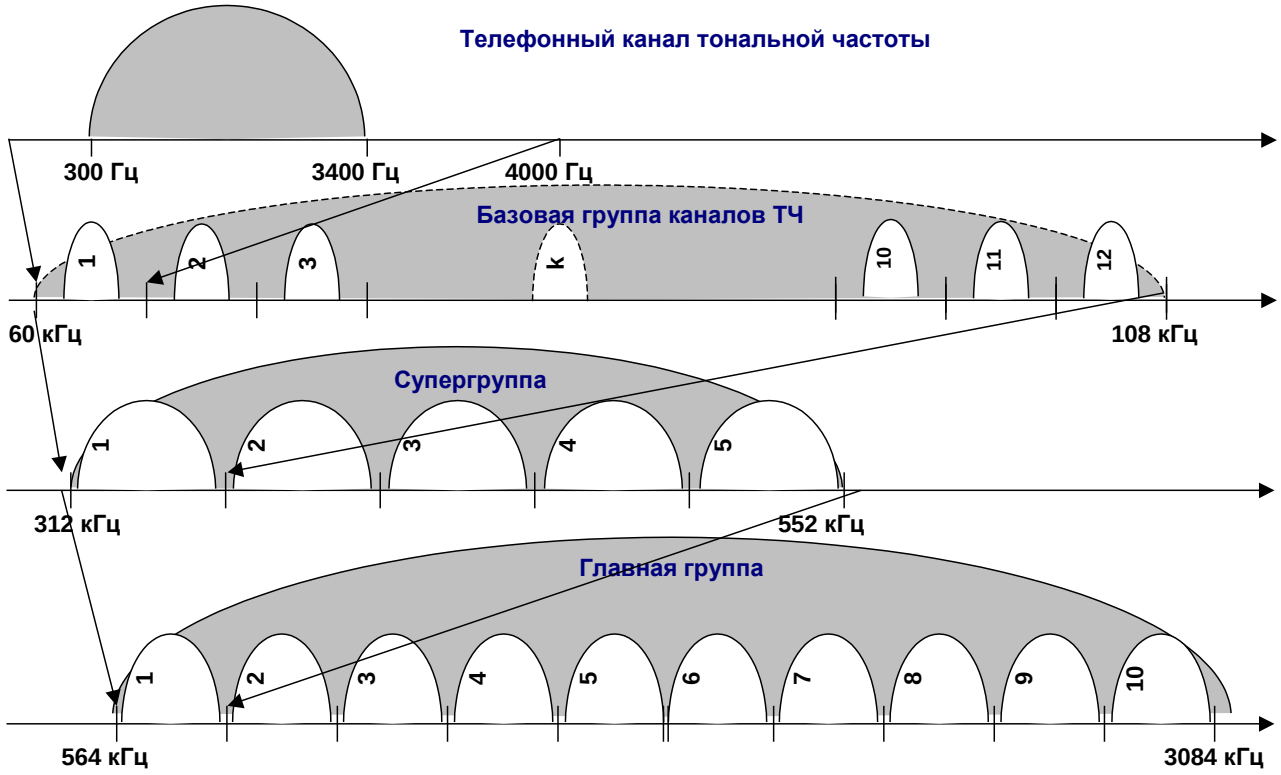
Метод с простоями является частным случаем метода скользящего окна, когда размер окна равен единице.

Метод скользящего окна имеет два параметра, которые могут заметно влиять на эффективность передачи данных между передатчиком и приемником — размер окна и величина тайм-аута ожидания квитанции.

Выбор тайм-аута зависит не от надежности сети, а от задержек передачи кадров сетью.

Во многих реализациях метода скользящего окна величина окна и тайм-аут выбираются адаптивно, в зависимости от текущего состояния сети.

Коммутация каналов



Частотное уплотнение телефонных каналов

5

Методы коммутации.

Любые сети связи поддерживают некоторый способ коммутации своих абонентов между собой. Невозможно предоставить каждой паре взаимодействующих абонентов свою собственную некоммутируемую физическую линию связи которой они могли бы монопольно «владеть» в течение длительного времени. Поэтому в любой сети всегда применяется какой-либо способ коммутации абонентов.

Абоненты соединяются с коммутаторами индивидуальными линиями связи, каждая из которых используется в любой момент времени только одним, закрепленным за этой линией абонентом. Между коммутаторами линии связи разделяются несколькими абонентами, то есть используются совместно.

Существуют три принципиально различные схемы коммутации абонентов в сетях:

- коммутация каналов,
- коммутация пакетов и
- коммутация сообщений.

Сети с коммутацией каналов имеют более богатую историю, они ведут свое происхождение от первых телефонных сетей. Сети с коммутацией пакетов сравнительно молоды, они появились в конце 60-х годов как результат экспериментов с первыми глобальными компьютерными сетями. Сети с коммутацией сообщений послужили прототипом современных сетей с коммутацией пакетов и сегодня они в чистом виде практически не существуют.

Каждая из этих схем имеет свои преимущества и недостатки, но по долгосрочным прогнозам многих специалистов будущее принадлежит технологии коммутации пакетов, как более гибкой и универсальной.

Как сети с коммутацией пакетов, так и сети с коммутацией каналов можно разделить на два класса по другому признаку — на сети с динамической коммутацией и сети с постоянной коммутацией.

В первом случае сеть разрешает устанавливать соединение по инициативе пользователя сети. Коммутация выполняется на время сеанса связи, а затем (опять же по инициативе одного из взаимодействующих пользователей) связь разрывается. В общем случае любой пользователь сети может соединиться с любым другим пользователем сети.

Пример - телефонные сети общего пользования, локальные сети, сети TCP/IP.

Во втором случае сеть не предоставляет пользователю возможность выполнить динамическую коммутацию с другим произвольным пользователем сети. Вместо этого сеть разрешает паре пользователей заказать соединение на длительный период времени. Соединение устанавливается не пользователями, а персоналом, обслуживающим сеть. Время, на которое устанавливается постоянная коммутация, измеряется обычно несколькими месяцами. Режим постоянной коммутации в сетях с коммутацией каналов часто называется сервисом выделенных (dedicated) или арендуемых (leased) каналов.

Пример - сети технологии SDH, на основе которых строятся выделенные каналы связи с пропускной способностью в несколько гигабит в секунду.

Некоторые типы сетей поддерживают оба режима работы. Например, сети X.25 и ATM могут предоставлять пользователю возможность динамически связаться с любым другим пользователем сети и в то же время отправлять данные по постоянному соединению одному вполне определенному абоненту.

Коммутация каналов.

Коммутация каналов подразумевает образование непрерывного составного физического канала из последовательно соединенных отдельных канальных участков для прямой передачи данных между узлами.

В настоящее время для мультиплексирования абонентских каналов используются две техники:

- техника частотного мультиплексирования (Frequency Division Multiplexing, FDM);
- техника мультиплексирования с разделением времени (Time Division Multiplexing, TDM).

Коммутация каналов на основе частотного мультиплексирования.

Техника частотного мультиплексирования каналов была разработана для телефонных сетей, но применяется она и для других видов сетей, например сетей кабельного телевидения.

Рассмотрим особенности этого вида мультиплексирования на примере телефонной сети.

Речевые сигналы имеют спектр шириной примерно в 10 000 Гц, однако основные гармоники укладываются в диапазон от 300 до 3400 Гц. Поэтому для качественной передачи речи достаточно образовать между двумя собеседниками канал с полосой пропускания в 3100 Гц, который и используется в телефонных сетях для соединения двух абонентов. В то же время полоса пропускания кабельных систем, соединяющих телефонные коммутаторы между собой, составляет сотни килогерц, а иногда и сотни мегагерц.

Для разделения абонентских каналов характерна техника модуляции высокочастотного несущего синусоидального сигнала низкочастотным речевым сигналом.

Коммутатор выполняет перенос частоты каждого канала в свой диапазон частот. Полосы делают шириной в 4 кГц, а не в 3,1 кГц, оставляя между ними страховой промежуток в 900 Гц. В канале между двумя FDM-коммутаторами одновременно передаются сигналы всех абонентских каналов, но каждый из них занимает свою полосу частот. Такой канал называют уплотненным.

В сетях на основе FDM-коммутации принято несколько уровней иерархии уплотненных каналов. Первый уровень уплотнения образуют 12 абонентских каналов, которые составляют **базовую группу** каналов, занимающую полосу частот шириной в 48 кГц с границами от 60 до 108 кГц. Второй уровень уплотнения образуют 5 базовых групп, которые составляют **супергруппу**, с полосой частот шириной в

240 кГц и границами от 312 до 552 кГц. Супергруппа передает данные 60 абонентских каналов тональной частоты. Десять супергрупп образуют **главную группу**, которая используется для связи между коммутаторами на больших расстояниях. Главная группа передает данные 600 абонентов одновременно и требует от канала связи полосу пропускания шириной не менее 2520 кГц с границами от 564 до 3084 кГц.

Коммутаторы FDM могут выполнять как динамическую, так и постоянную коммутацию. При динамической коммутации один абонент инициирует соединение с другим, посылая в сеть номер вызываемого абонента. Коммутатор динамически выделяет данному абоненту одну из свободных полос своего уплотненного канала. При постоянной коммутации за абонентом полоса в 4 кГц закрепляется на длительный срок путем настройки коммутатора.

Коммутация каналов на основе разделения времени.

Эта техника носит название мультиплексирования с разделением времени (TDM).

Аппаратура TDM-сетей — мультиплексоры, коммутаторы, демультиплексоры — работает в режиме разделения времени, поочередно обслуживая в течение цикла своей работы все абонентские каналы. Цикл работы оборудования TDM равен 125 мкс, что соответствует периоду следования замеров голоса в цифровом абонентском канале. Это значит, что мультиплексор или коммутатор успевает вовремя обслужить любой абонентский канал и передать его очередной замер далее по сети. Каждому соединению выделяется один квант времени цикла работы аппаратуры, называемый также тайм-слотом.

Мультиплексор принимает информацию по N входным каналам от конечных абонентов, каждый из которых передает данные по абонентскому каналу со скоростью 64 Кбит/с — 1 байт каждые 125 мкс. В каждом цикле мультиплексор выполняет следующие действия:

- прием от каждого канала очередного байта данных;
- составление из принятых байтов уплотненного кадра, называемого также обоймой;
- передача уплотненного кадра на выходной канал с битовой скоростью, равной Nx64 Кбит/с.

Порядок байт в обойме соответствует номеру входного канала, от которого этот байт получен. Количество обслуживаемых мультиплексором абонентских каналов зависит от его быстродействия. Например, мультиплексор T1, представляющий собой первый промышленный мультиплексор, работавший по технологии TDM, поддерживает 24 входных абонентских канала, создавая на выходе обоймы стандарта T1, передаваемые с битовой скоростью 1,544 Мбит/с.

Демультиплексор выполняет обратную задачу — он разбирает байты уплотненного кадра и распределяет их по своим нескольким выходным каналам, при этом он считает, что порядковый номер байта в обойме соответствует номеру выходного канала.

Коммутатор принимает уплотненный кадр по скоростному каналу от мультиплексора и записывает каждый байт из него в отдельную ячейку своей буферной памяти, причем в том порядке, в котором эти байты были упакованы в уплотненный кадр. Для выполнения операции коммутации байты извлекаются из буферной памяти не в порядке поступления, а в таком порядке, который соответствует поддерживаемым в сети соединениям абонентов. Так, например, если первый абонент должен соединиться со вторым абонентом в правой части сети, то байт, записанный в первую ячейку буферной памяти, будет извлекаться из нее вторым. «Перемешивая» нужным образом байты в обойме, коммутатор обеспечивает соединение конечных абонентов в сети.

Работа оборудования TDM напоминает работу сетей с коммутацией пакетов, так как каждый байт данных можно считать некоторым элементарным пакетом. Однако, в отличие от пакета компьютерной сети, «пакет» сети TDM не имеет индивидуального адреса. Его адресом является порядковый номер в обойме или номер выделенного тайм-слота в мультиплексоре или коммутаторе. Сети, использующие технику TDM, требуют синхронной работы всего оборудования. Нарушение синхронности разрушает требуемую коммутацию абонентов, так как при этом теряется адресная информация.

Существует модификация техники TDM, называемая статистическим разделением канала во времени. Эта техника разработана специально для того, чтобы с помощью временно свободных тайм-

слотов одного канала можно было увеличить пропускную способность остальных. Развитием идей статистического мультиплексирования стала технология асинхронного режима передачи — ATM, которая вобрала в себя лучшие черты техники коммутации каналов и пакетов.

Сети TDM могут поддерживать либо режим динамической коммутации, либо режим постоянной коммутации, а иногда и оба эти режима. Так, например, основным режимом цифровых телефонных сетей, работающих на основе технологии TDM, является динамическая коммутация, но они поддерживают также и постоянную коммутацию, предоставляя своим абонентам службу выделенных каналов.

Существует аппаратура, которая поддерживает только режим постоянной коммутации. К ней относится оборудование типа T1/E1. Такое оборудование используется для построения первичных сетей, основной функцией которых является создание выделенных каналов между коммутаторами, поддерживающими динамическую коммутацию.

Общие свойства сетей с коммутацией каналов.

Гарантированная пропускная способность сети после установления соединения.

Сети с коммутацией каналов хорошо приспособлены для коммутации потоков данных постоянной скорости, когда единицей коммутации является не отдельный байт или пакет данных, а долговременный синхронный поток данных между двумя абонентами.

В зависимости от направления возможной передачи данных способы передачи данных по линии связи делятся на следующие типы:

- симплексный — передача осуществляется по линии связи только в одном направлении;
- полудуплексный — передача ведется в обоих направлениях, но попеременно во времени. Примером такой передачи служит технология Ethernet;
- дуплексный — передача ведется одновременно в двух направлениях.

Дуплексный режим — наиболее универсальный и производительный способ работы канала. Самым простым вариантом организации дуплексного режима является использование двух независимых физических каналов (двух пар проводников или двух световодов) в кабеле, каждый из которых работает в симплексном режиме, то есть передает данные в одном направлении. Именно такая идея лежит в основе реализации дуплексного режима работы во многих сетевых технологиях, например Fast Ethernet или ATM.

Иногда такое простое решение оказывается недоступным или неэффективным. Чаще всего это происходит в тех случаях, когда для дуплексного обмена данными имеется всего один физический канал, а организация второго связана с большими затратами. Например, при обмене данными с помощью модемов через телефонную сеть у пользователя имеется только один физический канал связи с АТС — двухпроводная линия, и приобретать второй вряд ли целесообразно. В таких случаях дуплексный режим работы организуется на основе разделения канала на два логических подканала с помощью техники FDM или TDM.

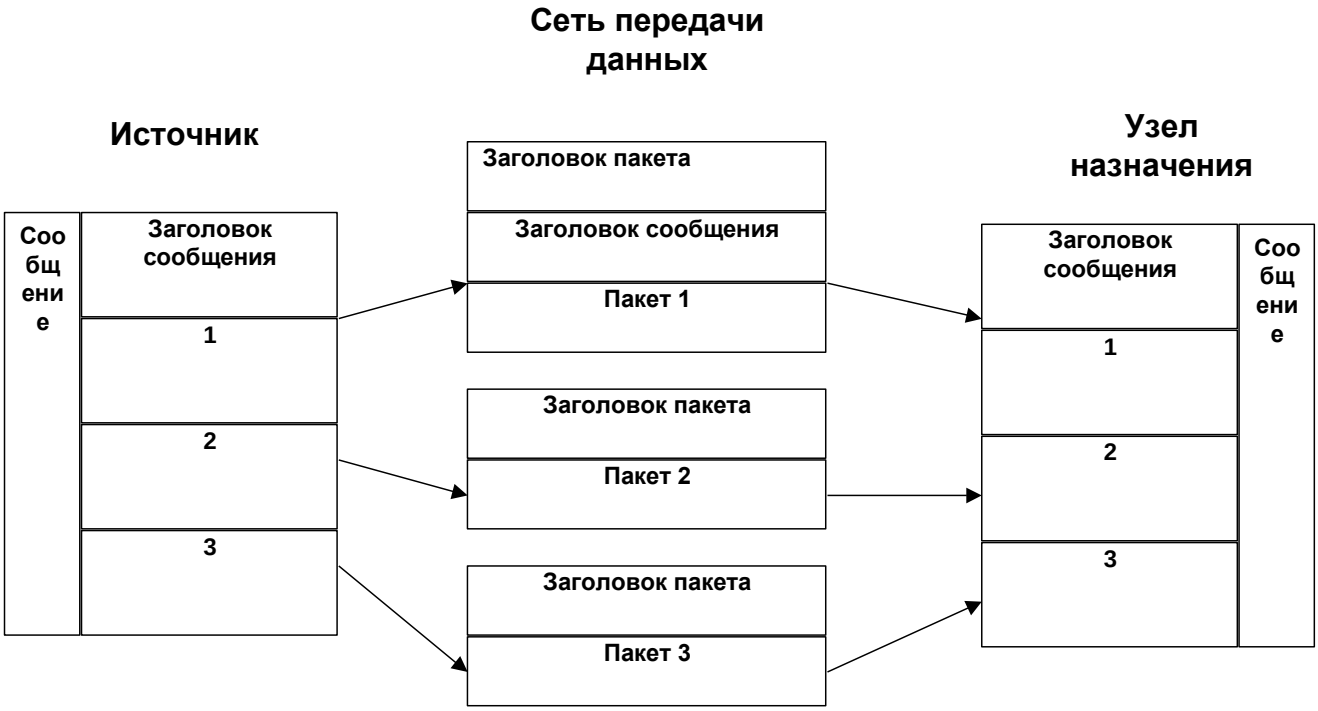
Модемы для организации дуплексного режима работы на двухпроводной линии применяют технику FDM. Модемы, использующие частотную модуляцию, работают на четырех частотах: две частоты — для кодирования единиц и нулей в одном направлении, а остальные две частоты — для передачи данных в обратном направлении.

При цифровом кодировании дуплексный режим на двухпроводной линии организуется с помощью техники TDM. Часть тайм-слотов используется для передачи данных в одном направлении, а часть — для передачи в другом направлении. Обычно тайм-слоты противоположных направлений чередуются, из-за чего такой способ иногда называют «пинг-понговой» передачей. TDM-разделение линии характерно, например, для цифровых сетей с интеграцией услуг (ISDN).

В волоконно-оптических кабелях при использовании одного оптического волокна для организации дуплексного режима работы применяется передача данных в одном направлении с помощью светового пучка одной длины волны, а в обратном — другой длины волны. Такая техника

относится к методу FDM, однако для оптических кабелей она получила название разделения по длине волны (WDM).

Коммутация пакетов



Разбиение сообщения на пакеты

6

Коммутация пакетов.

Принципы коммутации пакетов.

Коммутация пакетов — это техника коммутации абонентов, которая была специально разработана для эффективной передачи компьютерного трафика. Эксперименты по созданию первых компьютерных сетей на основе техники коммутации каналов показали, что этот вид коммутации не позволяет достичь высокой общей пропускной способности сети. Суть проблемы заключается в пульсирующем характере трафика, который генерируют типичные сетевые приложения. Например, при обращении к удаленному файловому серверу пользователь сначала просматривает содержимое каталога этого сервера, что порождает передачу небольшого объема данных. Затем он открывает требуемый файл в текстовом редакторе, и эта операция может создать достаточно интенсивный обмен данными, особенно если файл содержит объемные графические включения. После отображения нескольких страниц файла пользователь некоторое время работает с ними локально, что вообще не требует передачи данных по сети, а затем возвращает модифицированные копии страниц на сервер — и это снова порождает интенсивную передачу данных по сети.

Если для описанной сессии организовать коммутацию канала между компьютером пользователя и сервером, то большую часть времени канал будет простаивать. В то же время коммутационные возможности сети будут использоваться — часть тайм-слотов или частотных полос коммутаторов будет занята и недоступна другим пользователям сети.

При коммутации пакетов все передаваемые пользователем сети сообщения разбиваются в исходном узле на сравнительно небольшие части, называемые пакетами. Напомним, что сообщением

называется логически завершенная порция данных — запрос на передачу файла, ответ на этот запрос, содержащий весь файл, и т. п. Сообщения могут иметь произвольную длину, от нескольких байт до многих мегабайт.

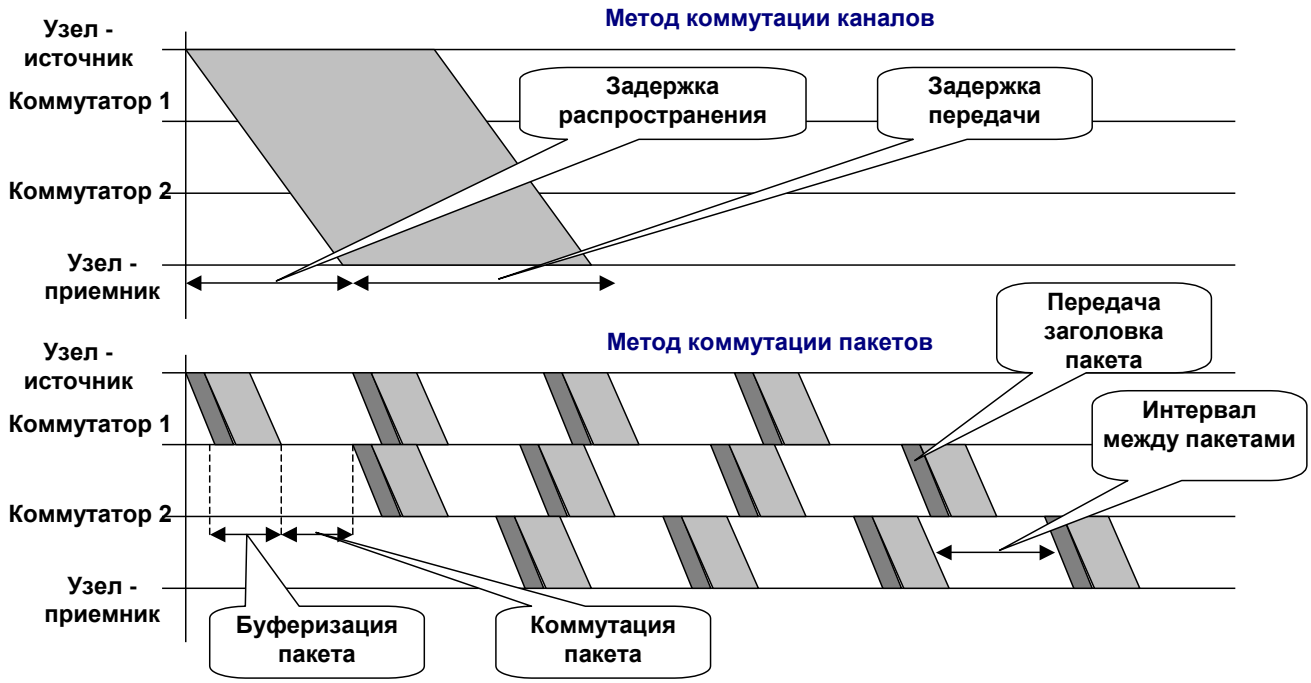
Напротив, пакеты обычно тоже могут иметь переменную длину, но в узких пределах, например от 46 до 1500 байт. Каждый пакет снабжается заголовком, в котором указывается адресная информация, необходимая для доставки пакета узлу назначения. Пакеты транспортируются в сети как независимые информационные блоки. Коммутаторы сети принимают пакеты от конечных узлов и на основании адресной информации передают их друг другу, а в конечном итоге — узлу назначения.

Для пары абонентов наиболее эффективным было бы предоставление им в единоличное пользование скоммутированного канала связи, как это делается в сетях с коммутацией каналов. При этом способе время взаимодействия этой пары абонентов было бы минимальным, так как данные без задержек передавались бы от одного абонента другому. Сеть с коммутацией пакетов замедляет процесс взаимодействия конкретной пары абонентов, так как их пакеты могут ожидать в коммутаторах, пока по магистральным связям передаются другие пакеты, пришедшие в коммутатор ранее.

Тем не менее общий объем передаваемых сетью компьютерных данных в единицу времени при технике коммутации пакетов будет выше, чем при технике коммутации каналов. Это происходит потому, что пульсации отдельных абонентов в соответствии с законом больших чисел распределяются во времени.

Поэтому коммутаторы постоянно и достаточно равномерно загружены работой, если число обслуживаемых ими абонентов действительно велико. Трафик, поступающий от конечных узлов на коммутаторы, очень неравномерно распределен во времени. Однако коммутаторы более высокого уровня иерархии загружены более равномерно, и поток пакетов в магистральных каналах, соединяющих коммутаторы верхнего уровня, имеет почти максимальный коэффициент использования. Более высокая эффективность сетей с коммутацией пакетов по сравнению с сетями с коммутацией каналов (при равной пропускной способности каналов связи) была доказана в 60-е годы как экспериментально, так и с помощью имитационного моделирования.

Коммутация пакетов



Задержки передачи данных в сетях с коммутацией каналов и коммутацией пакетов

7

Пропускная способность сетей с коммутацией пакетов.

Одним из отличий метода коммутации пакетов от метода коммутации каналов является неопределенность пропускной способности соединения между двумя абонентами. В методе коммутации каналов после образования составного канала пропускная способность сети при передаче данных между конечными узлами известна — это пропускная способность канала. Данные после задержки, связанной с установлением канала, начинают передаваться на максимальной для канала скорости (рис. вверху). Время передачи сообщения в сети с коммутацией каналов равно сумме задержки распространения сигнала по линии связи и задержки передачи сообщения. Задержка распространения сигнала зависит от скорости распространения электромагнитных волн в конкретной физической среде, которая колеблется от 0,6 до 0,9 скорости света в вакууме. Время передачи сообщения равно V/C, где V — объем сообщения в битах, а C — пропускная способность канала в битах в секунду.

В сети с коммутацией пакетов наблюдается принципиально другая картина.

Процедура установления соединения в этих сетях, если она используется, занимает примерно такое же время, как и в сетях с коммутацией каналов, поэтому будем сравнивать только время передачи данных.

На рис. внизу показан пример передачи в сети с коммутацией пакетов. Предполагается, что в сеть передается сообщение того же объема, однако оно разделено на пакеты, каждый из которых снабжен заголовком. При передаче этого сообщения возникают дополнительные временные задержки:

- задержки на передачу заголовком,
- задержки между пакетами (время формирования пакета стеком протоколов),

- задержки времени буферизации и коммутации в промежуточном оборудовании.

Основной недостаток сети с коммутацией пакетов – неопределенная пропускная способность сети. Это плата за ее общую эффективность. Аналогия – многозадачная ОС. Время выполнения определенного приложения оценить невозможно, т.к. оно зависит от количества других приложений.

Лекция 6. Управление сетевым трафиком.

Чтобы почувствовать актуальность темы – простой пример (Столлингс, «Современные компьютерные сети», глава 8). Веб-сервер обслуживает индивидуальные запросы в среднем за 1мс. Чтобы упростить задачу – ровно за 1мс. Если скорость поступления запросов – 1 запрос в миллисекунду – то кажется, можно утверждать, что сервер справится с такой нагрузкой.

1. Идеальный случай – постоянный интервал прихода запросов, сервер обрабатывает запрос сказу как он приходит. Как только завершается обработка – сразу переходит к следующему.

2. Более реалистичный вариант. Средняя скорость – такая-же, но она не постоянна. В течение одного интервала может не поступить ни одного запроса, а в другое время – большое количество. Но средняя скорость – постоянна. В период высокой нагрузки сервер хранит запросы в буфере, т.е. они стоят в очереди на обработку. Самый важный вопрос в данном случае – насколько большим должен быть буфер.

Есть таблицы, показывающие поведение такой системы:

Управление сетевым трафиком

запросов еще немного увеличена — до 99 % от производительности сервера, — в результате среднее количество запросов в буфере выросло до 2583. Таким образом, совсем незначительное увеличение средней скорости поступления запросов приводит к почти 40-процентному росту числа запросов, ожидающих в очереди.

Таблица 8.1. Поведение очереди при нормализованной скорости прибытия, равной 0,5

Время	Вход	Выход	Очередь
0	0	0	0
1	88	88	0
2	796	796	0
3	1627	1000	627
4	51	678	0
5	34	34	0
6	966	966	0
7	714	714	0
8	1276	1000	276
9	494	769	0
10	933	933	0
11	107	107	0
12	241	241	0
13	16	16	0
14	671	671	0
15	643	643	0
16	812	812	0
17	262	262	0
18	218	218	0
19	1378	1000	378
20	507	885	0
21	15	15	0
22	820	820	0
23	1253	1000	253
24	307	559	0
25	540	540	0
26	190	190	0
27	500	500	0
28	96	96	0
29	943	943	0
30	105	105	0
31	183	183	0
32	447	447	0
33	542	542	0
34	166	166	0
35	165	165	0
36	490	490	0

Время	Вход	Выход	Очередь
37	510	510	0
38	877	877	0
39	37	37	0
40	163	163	0
41	104	104	0
42	42	42	0
43	291	291	0
44	645	645	0
45	363	363	0
46	134	134	0
47	920	920	0
48	1507	1000	507
49	598	1000	105
50	172	277	0
Среднее	499	499	43

Таблица 8.2. Поведение очереди при нормализованной скорости прибытия, равной 0,95

Время	Вход	Выход	Очередь
0	167	167	0
2	1512	1000	512
3	3091	1000	2604
4	97	1000	1701
5	65	1000	765
6	1835	1000	1601
7	1357	1000	1957
8	2424	1000	3382
9	939	1000	3320
10	1773	1000	4093
11	203	1000	3269
12	458	1000	2754
13	30	1000	1784
14	1275	1000	2059
15	1222	1000	2281
16	1543	1000	2824
17	498	1000	2322
18	414	1000	1736
19	2618	1000	3354
20	963	1000	3317
21	29	1000	2346
22	1558	1000	2904
23	2381	1000	4285

продолжение

Первая таблица – предполагается, что в систему поступает в среднем 500 запросов в секунду, что

соответствует половине производительности сервера. Записи в таблице показывают количество запросов, прибывающих каждую секунду, количество запросов обработанных в течение этой секунды и количество избыточных запросов, поставленных в очередь. Данные в таблице содержатся за 50 секунд.

Среднее значение за 50 секунд – 43 запроса в очереди. Пиковое – 600.

Управление сетевым трафиком

Таблица 8.2 (*продолжение*)

Время	Вход	Выход	Очередь
24	583	1000	3868
25	1026	1000	3894
26	361	1000	3255
27	950	1000	3205
28	182	1000	2387
29	1792	1000	3179
30	200	1000	2378
31	348	1000	1726
32	849	1000	1575
33	1030	1000	1605
34	315	1000	921
35	314	1000	234
36	931	1000	165
37	969	1000	134
38	1666	1000	800
39	70	871	0
40	310	310	0
41	198	198	0
42	80	80	0
43	553	553	0
44	1226	1000	226
45	690	915	0
46	255	255	0
47	1748	1000	748
48	2863	1000	2611
49	1136	1000	2748
50	327	1000	2074
Среднее	948	907	1859

Таблица 8.3. Поведение очереди при нормализованной скорости прибытия, равной 0,99

Время	Вход	Выход	Очередь
0	0	0	0
1	174	174	0
2	1576	1000	576
3	3221	1000	2798
4	101	1000	1899
5	67	1000	966
6	1913	1000	1879
7	1414	1000	2292
8	2526	1000	3819
9	978	1000	3797

Время	Вход	Выход	Очередь
10	1847	1000	4644
11	212	1000	3856
12	477	1000	3333
13	32	1000	2365
14	1329	1000	2693
15	1273	1000	2967
16	1608	1000	3574
17	519	1000	3093
18	432	1000	2525
19	2728	1000	4253
20	1004	1000	4257
21	30	1000	3287
22	1624	1000	3910
23	2481	1000	5391
24	608	1000	4999
25	1069	1000	5068
26	376	1000	4445
27	990	1000	4435
28	190	1000	3625
29	1867	1000	4492
30	208	1000	3700
31	362	1000	3062
32	885	1000	2947
33	1073	1000	3020
34	329	1000	2349
35	327	1000	1676
36	970	1000	1646
37	1010	1000	1656
38	1736	1000	2392
39	73	1000	1465
40	323	1000	788
41	206	994	0
42	83	83	0
43	576	576	0
44	1277	1000	277
45	719	996	0
46	265	265	0
47	1822	1000	822
48	2984	1000	2805
49	1184	1000	2990
50	341	1000	2330
Среднее	988	942	2583

Второй случай. 95% производительности. В среднем – 950 запросов в секунду.

Среднее количество ожидающих в буфере запросов – 1859. Это может показаться удивительным. Скорость поступления запросов увеличилась менее чем в 2 раза, а средняя длина очереди – в 40 раз.

Третья таблица. Скорость постуления запросов – 99% производительности. Среднее количество запросов – 2583. Незначительное увеличение скорости заросов приводит к 40%-му росту очереди.

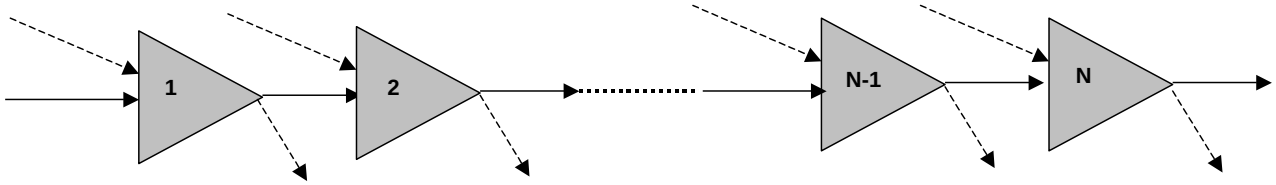
Этот грубый пример наводит на мысль о том, что поведение системы с очередью может не согласовываться с нашей интуицией.

Функции маршрутизаторов (коммутаторов):

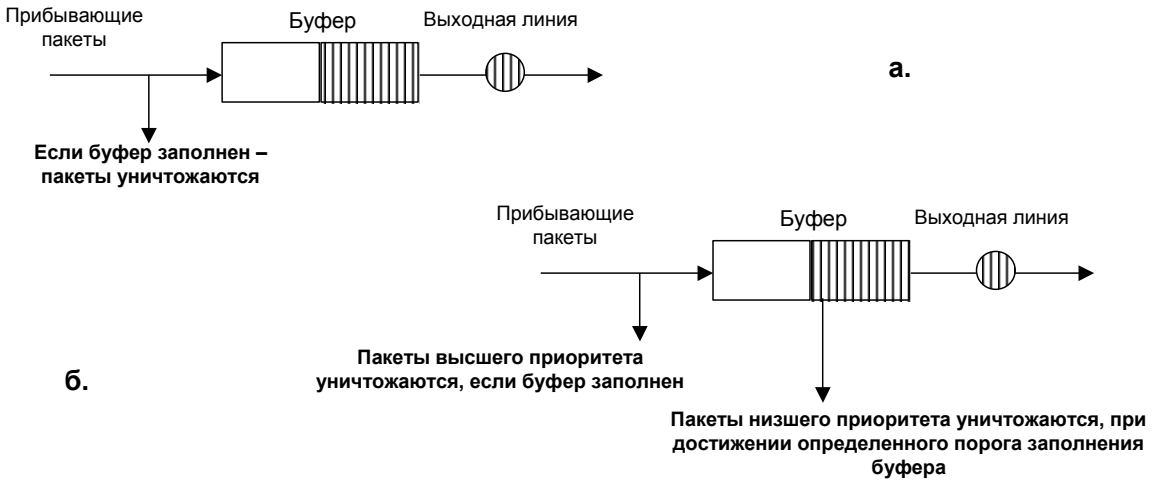
- демультиплексирование входного потока пакетов,
- определение для каждого пакета выходного порта и
- формирование посредством мультиплексирования выходного потока пакетов.

Таким образом, **ресурсами сети** являются буферная память и производительность процессора маршрутизатора, а также - пропускная способность линий передачи. Полный маршрут потока пакетов, в рамках конкретной прикладной задачи, или пакетов одного конкретного соединения через сеть, может быть представлен цепочкой мультиплексоров и линий передачи.

Управление сетевым трафиком



Модель полного пути пакета в форме цепочки мультиплексоров и линий передачи



Модель очереди FIFO без приоритезации (а) и с приоритезацией (б) 4

Пунктирными стрелками на этом рисунке представлены другие входные потоки. Эти потоки конкурируют между собой за ресурсы сети – емкость буферов, очередность обслуживания, и пропускную способность линий передачи.

Результаты этого соперничества

- крайне неравномерное распределение ресурсов сети
- снижение качества обслуживания одного, или всех потоков
- возникновение перегрузок на отдельных участках или даже коллапс сети (нулевая производительность).

Предупреждение нежелательных состояний сети требует мер управления сетевым трафиком, **целью** которых являются:

- предупреждение резкого падения производительности сети и
- обеспечение требуемого приложением уровня качества обслуживания (Quality of Service, QoS) генерируемого им потока.

Обеспечение качества обслуживания – комплексная задача и в ее решении принимают участие все уровневые протоколы, в том числе и сетевые. Качество обслуживания описывается системой параметров, специфической для каждого уровня. Параметры качества обслуживания для уровня сети:

- полное время доставки,
- неравномерность задержки доставки пакетов,
- уровень потерь пакетов.

Ясно, что значения параметров качества обслуживания потока в сети зависит от их значений, реализуемых в каждом узле коммутации. Так, полное время доставки (end-to-end delay) пакета определяется задержками, испытываемыми им на каждом коммутационном узле. Следовательно, и среднеяя задержка доставки – это сумма средних задержек. Очевидно, что если последние ограничены сверху, то максимальная полная задержка доставки пакетов данного потока не будет превосходить суммы верхних грананиц задержек на кажком узле коммутации.

Важной характеристикой качества обслуживания является джиттер (jitter) – т.е. величина неравномерности времени доставки пакетов на всем маршруте. Ее максимальное значение может вычисляться как разность верхней и нижней границ задержки доставки пакета.

Если пакет приходит в коммутационное устройство в момент, когда его буферная память полностью занята, то он просто уничтожается. Поэтотму, причинами потерь пакетов в коммуникационных устройствах являются также резкие скачки числа пакетов на входном порту и внезапное уменьшение пропускной способности выходной линии, или ее перегрузка. Поскольку подобные явления имеют случайный характер, то уровень потерь пакетов на маршруте от источника до приемника определяется суммой вероятностей их потерь на любом из коммутаторов, входящих в маршрут доставки.

Таким образом видно, что управление сетевым трафиком требует наличия:

- механизмов регулирования нагрузки, которые могут применяться к линиям и коммутационным устройствам в сети
- механизмов формирования (профилирования) потоков на входе в сеть
- механизмов справедливого распределения ресурсов сети, выделяемых для обслуживания разных потоков
- средств реализации сетевых политик (набора административных правил доступа к ресурсам сети).

Базовыми элементами реализации перечисленных механизмов являются

- алгоритмы обслуживания очередей пакетов и
- алгоритмы управления средней и пиковой скоростью передачи потоков.

FIFO и приоритетные очереди

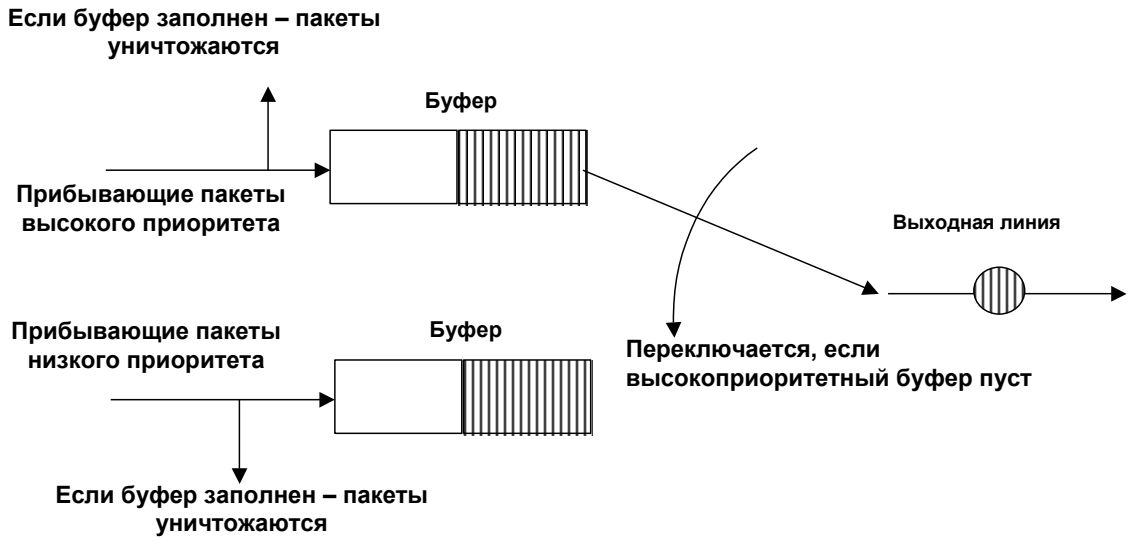
Механизм планирования обслуживания очередей (планировщик очередей) определяет порядок обслуживания пакетов и, следовательно, самым непосредственным образом влияет на распределение ресурсов узла коммутации. Планировщик очередей определяет сколько пакетов из каждого информационного потока будет обслужено в единицу времени.

Простейшим механизмом управления потоком пакетов является очередь с дисциплиной «Первый пришел – первый вышел» (First-In, First-Out, FIFO). В этом случае все прибывающие пакеты помещаются в одну общую очередь и обрабатываются последовательно в порядке прибытия *(Слайд 2, рисунок внизу, а)*.

Пакеты, приходящие в моменты полного заполнения буферной памяти, уничтожаются. Задержка обработки и уровень потерь пакетов в FIFO-буфере зависят от скорости их поступления и длины; поэтому с ростом скорости поступления пакетов и увеличением неравномерности их размера качество обслуживания пакетов в такой очереди будет ухудшаться. Очевидно также, что в такой очереди невозможно обеспечить разные уровни обслуживания пакетов, принадлежащих разным инфрмационнвм потокам. С дисциплиной FIFO связана еще одна проблема – монополизация ресурсов маршрутизатора потоком высокой интенсивности, который заполняет входной буфер и делает недоступными ресурсы маршрутизатора для всех остальных потоков.

Очередь FIFO может быть модифицирована так, что она сможет обеспечивать различный уровень обслуживания разным информационным потокам, т.е. очередь становится приоритетной. Этот механизм предполагает разделение входного трафика на классы (может быть, по типу протокола, по адресу источника, по типу приложения и т.п.) и тогда при достижении определенного порога загруженности буфера пакеты более низкого приоритета будут уничтожаться, сохраняя тем самым возможность обработки пакетов более высокого приоритета *(рисунок внизу, б)*. Последние будут обрабатываться до момента полной загруженности буфера.

Управление сетевым трафиком



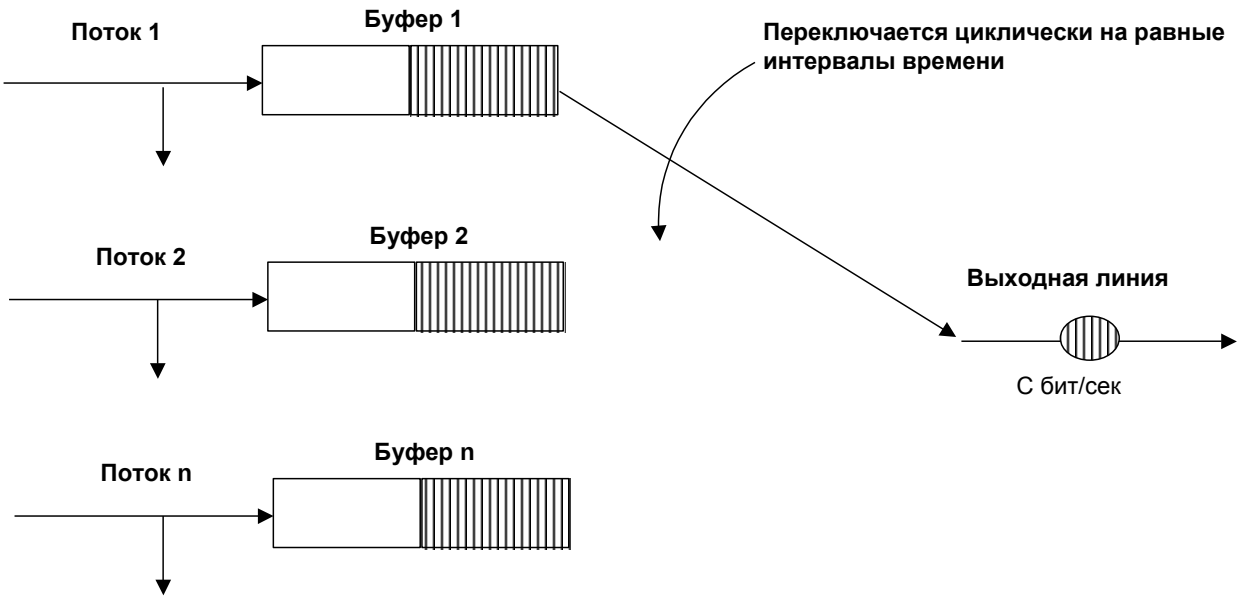
Модель очереди FIFO с приоритезацией на основе входных буферов

5

Другим вариантом решения задачи дифференциорования классов обслуживания пакетов является организация нескольких входных очередей (по одной для каждого класса обслуживания), конкурирующих за доступ к ресурсам (CPU и выходная линия) маршрутизатора. Как показано на слайде, выходная линия в любой момент является доступной для пакетов из буфера, соответствующего самому высокому классу обслуживания (например, пакетам потока, чувствительного к величине задержек) и лишь когда он пуст – обслуживается буфер низшего приоритета. Размер буферов в такой архитектуре также может быть различным, что позволяет удовлетворить разные требования к уровню вероятности потери пакетов.

Ресурсы коммутатора всегда ограниченны и ситуация, когда разные информационные потоки вынуждены конкурировать за доступ к ним, является типичной. При этом, рассмотренные выше варианты обеспечения разных классов обслуживания не решают проблему «справедливого» распределения ресурсов маршрутизатора. Действительно, при достаточно высоком трафике высшего приоритета потоки низшего приоритета могут вообще не получить доступа к ресурсам маршрутизатора; не решается также проблема распределения ресурсов между потоками одного класса сервиса, что приводит к монополизации их (ресурсов) более «жадными» приложениями.

Управление сетевым трафиком



Модель «справедливой» очереди

6

«Справедливая» очередь

«Справедливая очередь» нацелена на преодоление отмеченной выше ассиметрии распределения ресурсов коммутатора. Пусть каждый информационный поток обслуживается своей логической очередью. В идеальной системе (поток входных данных рассматривается как некая непрерывная и всегда существующая субстанция) пропускная способность выходной линии (C) может быть разделена поровну между всеми очередями посредством их циклического подключения к выходной линии на строго определенные и равные промежутки времени (см. слайд). При наличии *n* очередей, каждая из них получает возможность передавать свои пакеты в выходную линию со скоростью *C/n* бит/сек.

Эффективная скорость передачи может быть и выше, поскольку некоторые очереди становятся время от времени пустыми, и в это время другие из них могут передавать данные. Такая организация очереди предупреждает монополизацию ресурсов выходной линии «жадными» приложениями. Размер буфера для каждого потока может быть выбран таким, чтобы удовлетворялись специфические требования к уровню потерь пакетов конкретного потока.

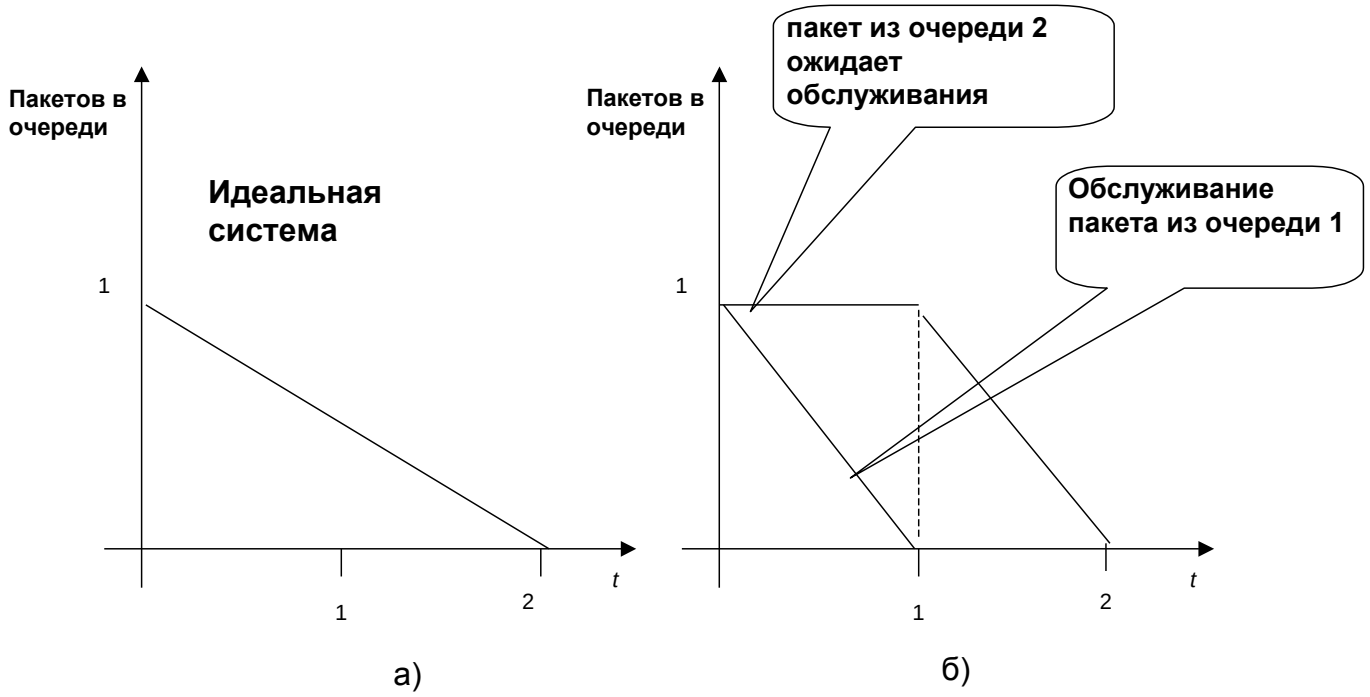
Однако, рассматриваемая очередь является «справедливой» когда мы считаем входные потоки идеальными (непрерывными). В этом случае, пропускная способность передающей линии действительно равномерно распределяется между всеми непустыми очередями. Однако, в силу дискретной природы потока и различия размеров его пакетов, «справедливость» не обеспечивается.

Каждая из входных очередей циклически получает доступ к выходной линии на строго определенные и равные промежутки времени, и за этот интервал может быть передано строго определенное количество бит информации. Если длина пакета превосходит это значение, то он не будет передан; если она меньше – то часть времени доступа к линии этой очередью будет потрачена непроизводитеьно.

Для уменьшения таких непроизводительных потерь можно дефрагментировать приходящие пакеты на относительно короткие составляющие, что, очевидно, будет вести к усложнению процедуры коммутации.

Посмотрем на различия в обслуживании «идеального» потока и потока пакетов конечной длины в системе с двумя очередями.

Управление сетевым трафиком



Обслуживание «справедливой» очереди в идеальной системе (а) и в системе с дискретным характером входного потока

7

Каждая из двух очередей содержит только по одному пакету размером *L* бит; при этом пропускная способность выходной линии составляет *L* бит/сек = 1 пакет/сек. В идеальной системе можно обеспечить для каждой очереди возможность передавать свои данные на скорости ½ пакета в секунду (рис. а).

Если обслуживание очередей организовано побитно (т.е. из каждой очереди равномерно передается по одному биту), то время цикла передачи составляет *2* секунды. Весь пакет из первой очереди будет передан за время *2-1/L* секунд, а передача пакета из второй очереди завершится ровно через 2 секунды (уже видна некоторая «несправедливость» в обслуживании).

Если же обслуживание очередей организовано на «пакетном» принципе (рис. б), то пакет из первой очереди будет обслужен за 1 секунду, а из второй – через 2 секунды («несправедливость» стала еще большей). Естественно, усредненная «несправедливость» будет тем меньше, чем больше пакетов будет обслужено из каждой очереди.

Однако, в случае неравенства длин пакетов распределение пропускной способности выходной линиии не будет «справедливым». Например, если длины пакетов первой очереди вдвое больше длин пакетов второй очереди, то при достаточной длине очередей и обслуживании их на «попакетной» основе эффективная полоса пропускания на данном коммутаторе для первого потока будет вдвое больше, чем для второго. Поэтому, разумным решением является предоставление каждому потоку такой доли ресурса выходной линии, чтобы время завершения обработки пакетов каждого из них оказалось близким к значению, которое было бы получено в идеальной системе с непрерывным

потоком входных данных. При попакетной дисциплине обслуживания единственной возможностью достижения этого остается выбор очередности обслуживания буферов.

Для этого каждый пакет, находящийся в очереди, следует снабдить *меткой времени окончания его обслуживания*, вычисленной исходя из модели непрерывного потока. Очередность же обслуживания входных буферов в каждом цикле определяется значениями этих меток по принципу «Пакет с меньшим значением метки обслуживается первым». Такую дисциплину обслуживания можно назвать **«попакетная справедливая» очередь**. Рассмотрим процедуру вычисления метки.

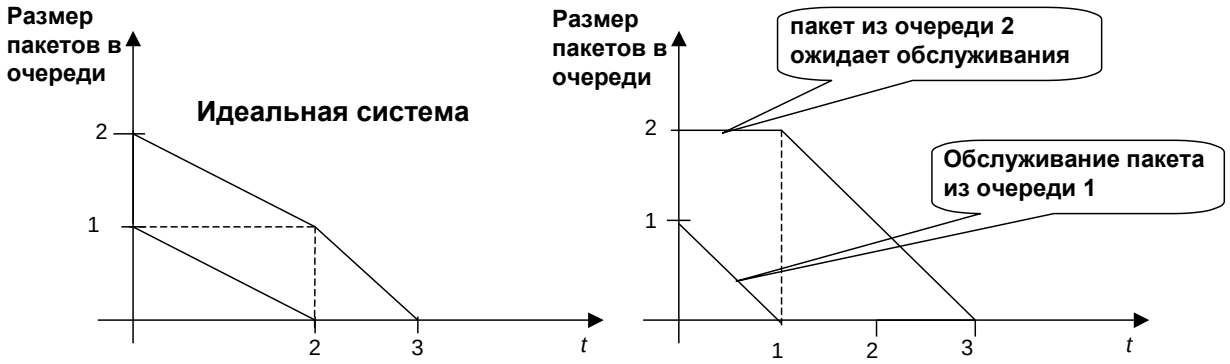
Управление сетевым трафиком

R(t)* – число циклов обслуживания *n* очередей, реализованных за время *t

t_k^i **- момент прибытия *k-го* пакета *i-го* потока в пустой буфер,** ***P(i,k)*** **- длина этого пакета (бит)**

$$R(t^*) = F(i,k) = R(t_k^i) + P(i,k) \qquad F(i,k) = F(i,k-1) + P(i,k)$$

***F(i,k)* - финишная метка обслуживания пакета**



Обслуживание «справедливой» очереди в идеальной системе и в системе с попакетной дисциплиной при разных длинах пакетов **8**

Пусть существуют *n* потоков, каждый из которых обслуживается отдельной очередью со скоростью 1 бит/сек (модель, максимально близкая к непрерывному потоку). Циклом обслуживания назовем период времени, в течении которого каждая очередь обслуживается по одному разу и *R(t)* – число циклов обслуживания *n* очередей, реализованных за время *t*. Будем считать, что *k-й* пакет *i-го* потока прибывает в пустой буфер своей очереди в момент t_k^i и длина этого пакета равна *P(i,k)* бит. Этот пакет на побитовой основе будет обработан за *P(i,k)* циклов, т.е в момент t^*, когда

$$R(t^*) = F(i,k) = R(t_k^i) + P(i,k).$$

Если пакет прибывает не в пустой буфер, то $F(i,k) = F(i,k-1) + P(i,k)$.

Величина *F(i,k)* и будет использована как финишная метка обслуживания пакета.

Величина *F(i,k)* не опреднляет действительное (реальное) время завершения обработки *k-го* пакета; она служит лишь признаком очередности, в которой будут обслуживаться буферы в пределах данного цикла.

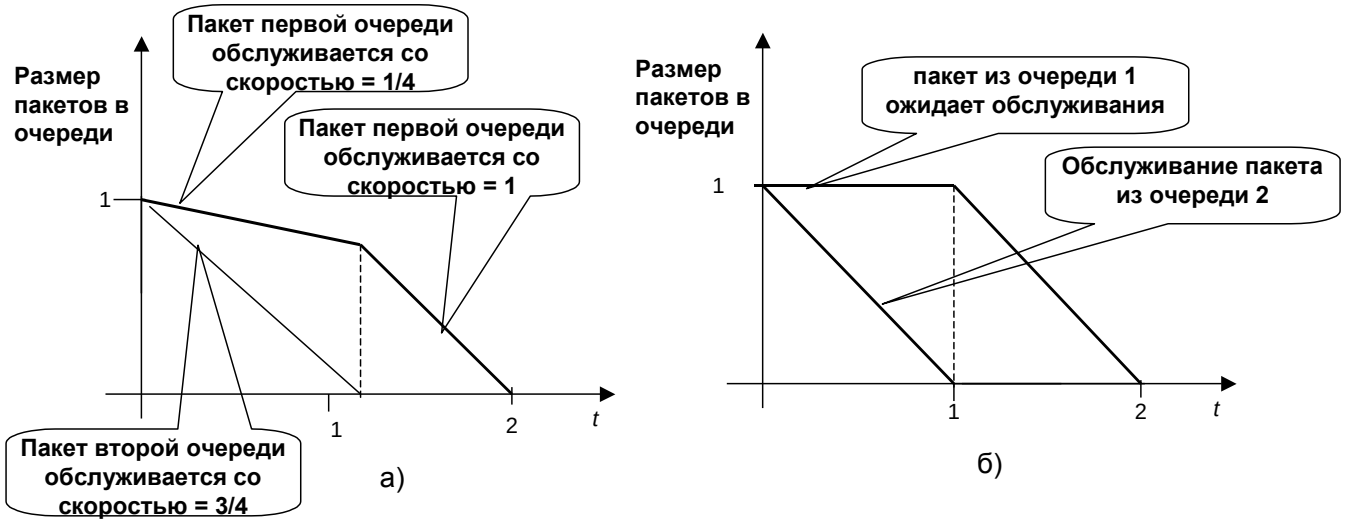
Для примера рассмотрим систему из двух очередей, первая из которых имеет один пакет единичной длины, а вторая – один пакет удвоенной длины.

В идеальной системе обслуживание очередей производится со скоростью пакет в ½ сек в течении времени пока обе они имеют данные (т.е. на интервале от 0 до 2 единиц времени) и со скоростью пакет в 1 сек - на интервале от 2 до 3 единиц времени (рис. слева).

В «попакетной справедливой» очереди значения финишних меток будут $F(1,1)= R(0)+ 1=1$ и $F(2,1)= R(0)+ 2=2$. Поскольку $F(1,1)< F(2,1)$, то обслуживание начинается с первой очереди (рис. справа).

Таким образом, если в идеальной системе поток с короткими пакетами передавался с **эффективной** скоростью ½ сек$^{-1}$., а поток с более длинным пакетом – со скоростью 2/3 сек$^{-1}$, то обслучивание их в «справедливой» очереди реализовывалось с эффективными скоростями равными 1сек$^{-1}$ и 2/3 сек$^{-1}$ соответственно. Если бы обслуживание началось с очереди, имеющей более длинный пакет, то эти скорости имели бы значения 1сек$^{-1}$ для «длинной» очереди и 1/3 сек$^{-1}$ для короткой. Последние значения отличаются от идеальных в большей степени.

Управление сетевым трафиком



Обслуживание «справедливой» взвешенной очереди в идеальной системе и в системе с попакетной дисциплиной при равных длинах пакетов

Мера значимости отражается показателем веса потока *Wi*

W1=1, W2=3

$$F(i,k) = \max\{F(i,k-1), R(t_k^i)\} + P(i,k)/W_i$$

$$F(1,1) = R(0) + 1/1 = 1 \qquad F(2,1) = R(0) + 1/3 = 1/3.$$

9

Взвешенная «справедливая» очередь

Определение взвешенной справедливости обобщает рассмотренный механизм «справедливого» распределения ограниченного ресурса на случай когда (потоки) «платят» разную стоимость за обладание единицей ресурса, что соответствует **разной значимости потоков**.

Мера значимости отражается показателем веса потока W_i . Пусть, как и выше, каждый информационный поток обслуживается отдельной очередью, но $W_1=1$, а $W_2=3$. Общая цена, которую готовы «заплатить» за ресурс маршрутизатора равна *4*. В этом случае «справедивость» заключается в том, что первый поток должен получить ¼ , а поток 2 – ¾ пропускной способности выходной линии.

Рисунока иллюстрирует обслуживание этих очередей в идеальной системе при условии, что длины очередей одинаковы. При обслуживании очередей на «побитовой» основе первая из них получает возможность в каждом цикле передать 1 бит, а вторая – 3 бита.

В системе с попакетной дисциплиной обслуживания «взвешенная справедливость» также может быть реализована. Пусть имеются n информационных потоков и «вес» каждого W_i. Значения финишных меток при этом вычисляются в соответствии с выражением

$$F(i,k) = \max\{ F(i,k-1), R(t_k^i)\} + P(i,k)/W_i$$

Последнее слагаемое в этом выражении показывает, что чем выше вес потока, тем меньшее значение финишной метки имеют его пакеты и тем быстрее они будут поступать в выходной канал коммутационного устройства.

Рисунок иллюстрирует обработку потоков с обозначенными выше весами в очередях с взвешенной справедливостью при попакетной дисциплине их обслуживания.

Очевидно, что $F(1,1) = R(0) + 1/1 = 1$ и $F(2,1) = R(0) + 1/3 = 1/3.$ Следовательно, пакет из второй очереди будет обслужен первым.

В заключение: взвешенная справедливая очередь является одним из инструментов обеспечения заданного уровня качества обслуживания в сети. Например, в сети Интернет существуют три стратегии обслуживания потоков.

- Первая из них реализует идею «как получится» и для нее достаточно применения FIFO-очередей.
- Вторая базируется на модели дифференциального обслуживания и предполагает разделение всего трафика на несколько классов, качество обслуживания каждого из которых удовлетворяет определенным ограничениям. При этом используется дисциплина приоритетных очередей. Эта стратегия относительно проста в реализации, но не гарантирует строгого обеспечения значений параметров качества обслуживания.
- Третий подход базируется на модели интегрального обслуживания, которая предполагает резервирование ресурсов, необходимых для обслуживания заданного потока с заданным качеством, на каждом маршрутизаторе от источника до приемника. Именнно здесь дисциплина взвешенных справедливых очередей находит свое практическое воплощение.

Лекция 7. Управление сетевым трафиком.

Управление сетевым трафиком

Интегрированные службы. Предоставляют коллективные услуги. Изучаются суммарные требования трафика.
- трафик ограничивается объемами, соответствующими текущим возможностям сети;
- резервируются ресурсы домена для предоставления определенного качества обслуживания в соответствии с требованиями.

Дифференцированные службы.
- не рассматриваются суммарные запросы трафика;
- не резервируются заранее сетевые ресурсы;
- трафик классифицируется по группам.

2

Два стандарта, представляющих собой взаимодополняющие структуры управления трафиком:

1. **Интегрированные службы**. Предоставлют интегрированные, или коллективные, услуги в данном домене (то есть части Интернета, в которой реализована определенная архитектура интегрированных служб). Изучаются суммарные требования трафика и,
2. во-первых, трафик ограничивается объемами, соответствующими текущим возможностям сети,
3. во-вторых, резервируются ресурсы домена для предоставления определенного качества обслуживания в соответствии с требованиями.
4. **Дифференцированные службы**. Наоборот, не рассматриваются суммарные запросы трафика, а также не резервируются заранее сетевые ресурсы. Вместо этого трафик классифицируется по группам. Каждая группа соответствующим образом помечается, а услуга, предоставляемая сетевыми элементами, зависит от членства в той или иной группе.

Т.е., можно сказать, что интегрированные услуги (т.е. совмещенные услуги) – это работа всей сети, которая анализирует свое состояние и рассматривает суммарные запросы, а дифференцированные услуги (раздельные) – предоставляются на основании информации внутри трафика (приоритеты и т.д.)

Архитектура интегрированных служб RFC16331

•Эластичный трафик
приспосабливается к изменениям задержки и пропускной способности, продолжая удовлетворять потребности приложения

•Неэластичный трафик
плохо приспосабливается к изменениям задержки и пропускной способности сети.

Требования к сети:
•Пропускная способность – необходим определенный минимум.
•Задержка
•Флуктуация – диапазон изменения задержек.
•Потеря пакетов – некоторые приложения реального времени допускают потерю пакетов.

Архитектура интегрированных служб

Вначале IP-сети могли предоставлять всем приложениям простую услугу по доставке пакетов с максимальными усилиями (то есть по остаточному принципу). Хотя заголовок протокола IPv4 содержит поля уровня приоритета и типа услуги, как правило, эта информация игнорируется маршрутизаторами.

Мультимедийные приложения, приложения групповой рассылки или приложения реального времени недостаточно хорошо поддерживаются такой конфигурацией.

Единственной сетевой схемой, с первого дня предназначенной для поддержки как традиционного TCP- и UDP-трафика, так и трафика реального времени, является сеть ATM. Однако установка сети ATM означает либо создание второй инфраструктуры для трафика реального времени, либо замену существующей IP-конфигурации ATM-структурой, при этом для реализации обоих вариантов требуются большие затраты.

Таким образом, в архитектуре TCP/IP существует необходимость в поддержании разных видов трафика с разными уровнями качества обслуживания. Основное требование заключается в добавлении маршрутизаторам новой функциональности. Для этого разрабатывается набор стандартов под общим названием **архитектура интегрированных служб.**

В общих чертах - RFC 16331, детали уточняются в других документах.

Трафик в объединенных сетях

Трафик в сети можно разделить на две большие категории: эластичный и неэластичный.

Эластичный трафик

Трафик, способный приспосабливаться к изменениям задержки и пропускной способности, продолжая удовлетворять потребности приложения. Это традиционный тип трафика, поддерживаемый IP-сетями, и именно для такого трафика разрабатывались объединенные сети.

Приложения, создающие подобный трафик, в качестве транспортного протокола, как правило, используют протокол TCP или UDP. В случае протокола UDP приложение расходует столько ресурсов, сколько имеется в наличии, вплоть до скорости, с которой приложение генерирует данные. В случае протокола TCP приложение расходует столько ресурсов, сколько имеется в наличии, вплоть до максимальной скорости, с которой оконечный получатель способен принимать данные.

К эластичным приложениям относятся приложения, работающие с помощью протоколов TCP и UDP, включая передачу файлов (FTP), электронную почту (SMTP), удаленную регистрацию (TELNET), управление сетью (SNMP) и доступ к www (HTTP).

Требования, предъявляемые данными приложениями, различаются, например:

▪ Электронная почта, как правило, нечувствительна к изменениям задержки.

▪ Интерактивные приложения, такие как удаленная регистрация и доступ к www, обладают высокой чувствительностью к задержке.

Неэластичный трафик

Плохо приспосабливается к изменениям задержки и пропускной способности сети.

Пример – realtime приложения.

Требования к сети:

▪ **Пропускная способность** – необходим определенный минимум ПС.

▪ **Задержка**. Пример – игра на бирже. Для того, чтобы не было запаздывания в действиях, необходима минимальная задержка.

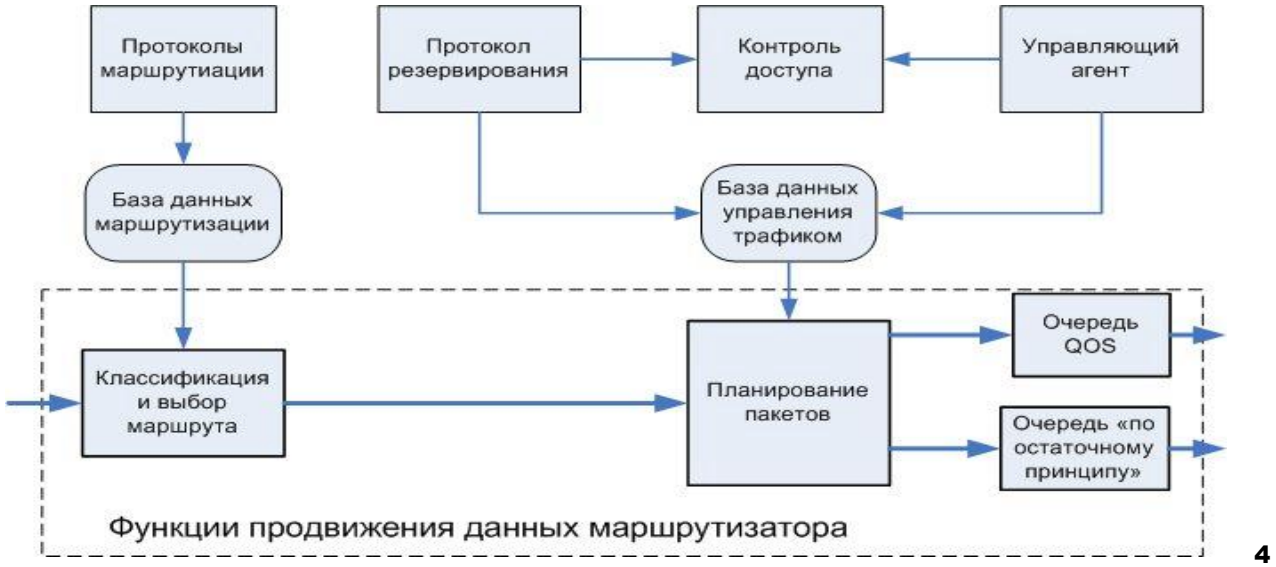
▪ **Флуктуация** – диапазон изменения задержек.

▪ **Потеря пакетов** – некоторые приложения реального времени допускают потерю пакетов.

Архитектура интегрированных служб

Функции архитектуры ISA:

- **Контроль допуска**
- **Алгоритм маршрутизации**
- **Дисциплина очередей**
- **Политика отбрасывания пакетов**



4

Функции архитектуры ISA

Некоторые **функции архитектуры ISA**, ориентированные на борьбу с перегрузкой и предоставление транспортных услуг с различными уровнями качества:

▪ **Контроль допуска**. Для транспорта с гарантированным уровнем качества обслуживания (в отличие от транспорта, обслуживаемого по остаточному принципу) архитектура ISA требует резервирования ресурсов для нового потока. Если маршрутизаторы совместно придут к соглашению о недостаточности ресурсов для гарантий запрашиваемого уровня качества обслуживания, тогда потоку отказывается в доступе.

▪ **Алгоритм маршрутизации**. Решение о выборе маршрута может приниматься на основе различных параметров качества обслуживания, а не только минимального значения задержки.

▪ **Дисциплина очередей**. Учитывает требования различных потоков.

▪ **Политика отбрасывания пакетов**. Политика очередей определяет, какой пакет будет передан следующим, если несколько пакетов стоят в очереди к одному и тому же выходному порту. Отдельным вопросом является выбор отбрасываемого пакета. Политика отбрасывания пакетов может быть важным элементом борьбы с перегрузкой и обеспечения гарантий качества обслуживания.

Компоненты архитектуры ISA

Общая схема реализации архитектуры ISA на маршрутизаторе. Отдельно выделены функции продвижения данных маршрутизатора. Эти функции выполняются для каждого пакета, и поэтому они должны быть тщательно оптимизированы. Остальные функции являются вспомогательными.

▪ **Протокол резервирования.** Этот протокол используется между маршрутизаторами и между маршрутизаторами и оконечными системами в целях резервирования ресурсов для нового потока и данного уровня качества обслуживания. Протокол резервирования - RSVP. Протокол резервирования обновляет **базу данных управления трафиком** используемую **функцией планирования пакетов** для определения услуг, предоставляемых пакетам каждого потока.

▪ **Контроль доступа**. Когда запрашивается новый поток, протокол резервирования вызывает функцию контроля доступа. Эта функция определяет, достаточно ли имеется ресурсов для нового потока с запрашиваемым уровнем качества обслуживания. Принятие решения основывается на текущем уровне обязательств по отношению к другим потокам и/или на текущем уровне нагрузки в сети.

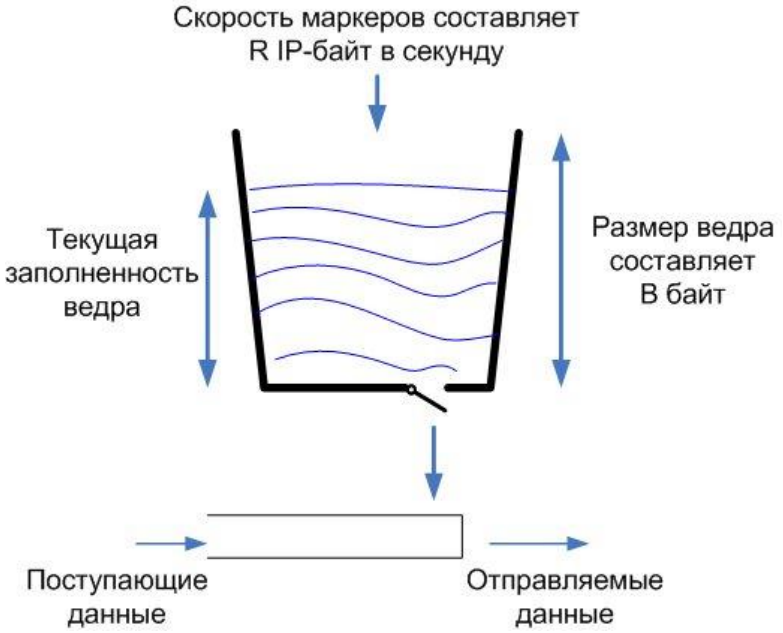
▪ **Управляющий агент** способен модифициро вать базу данных управления трафиком и управлять модулем контроля до ступа в соответствии с политикой контроля доступа.

▪ **Протокол маршрутизации** отвечает за поддержание базы данных маршрутизации, в которой для каждого адреса получателя и каждого потока содержатся данные о следующем ретрансляционном участке. Эти вспомогательные функции поддерживают основную задачу маршрутизации - продвижение пакетов. Следующие две принципиально основные функции осуществляют продвижение пакетов:

- **Классификация и выбор маршрута**. Входящие пакеты должны классифицироваться. Класс может соответствовать отдельному потоку или набору потоков с одинаковыми требованиями к качеству. Например, пакеты всех потоков с видеоданными или все потоки, относящиеся к одной организации, могут обрабатываться одинаково в плане распределения ресурсов и дисциплины очередей. Выбор класса основывается на полях в заголовках IP-пакетов. Зная класс пакета и IP-адрес получателя, функция классификации и выбора маршрута определяет адрес следующего ретрансляционного участка для этого пакета.
- **Планирование пакетов**. Функция планирования пакетов управляет одной или несколькими очередями для каждого выходного порта. Она определяет порядок, в котором передаются стоящие в очереди пакеты, а также, при необходимости, выбирает пакеты, выбрасываемые из очереди. Решения принимаются на основе класса пакета, содержимого базы данных управления трафиком, а также текущей и прошлой активности этого выходного порта. **Например**: часть задачи планирования - определение, превышает ли трафик пакетов данного потока запрошенный уровень пропускной способности, и, если превышает, решение, как поступить с лишними пакетами.

Архитектура интегрированных служб

Три категории обслуживания:
- **гарантированное обслуживание;**
- **обслуживание с контролируемой нагрузкой;**
- **обслуживание с максимальными усилиями (то есть по остаточному принципу).**



5

Службы архитектуры ISA

Службы ISA для потока пакетов определены на двух уровнях. **Во-первых**, службы предоставляют ряд общих категорий обслуживания (чуть ниже), каждая из которых обеспечивает определенный обобщенный тип гарантий обслуживания. **Во-вторых**, в каждой категории обслуживание конкретного потока определяется заданными параметрами. Вместе эти параметры называются **спецификацией трафика**.

Три категории обслуживания:

- гарантированное обслуживание;
- обслуживание с контролируемой нагрузкой;
- обслуживание с максимальными усилиями (то есть по остаточному принципу).

Приложение может потребовать зарезервировать для потока ресурсы с гарантированным или контролируемым уровнем качества. При этом точные параметры указываются в спецификации трафика. Если запрос на резервирование принимается, тогда спецификация трафика становится частью договора между потоком данных и службой. Служба соглашается предоставлять обслуживание с запрашиваемым уровнем качества до тех пор, пока поток данных соответствует спецификации трафика. Пакетам, превышающим параметры, указанные в спецификации, по умолчанию предоставляется обслуживание по остаточному принципу.

Прежде чем перейти к рассмотрению категорий служб архитектуры ISA, следует определить одну общую концепцию: спецификацию трафика **маркерного ведра**. В контексте архитектуры интегрированных служб этот способ описания трафика обладает тремя преимуществами:

- Большое количество источников трафика могут быть просто и точно описаны с помощью схемы маркерного ведра.
- Схема маркерного ведра предоставляет лаконичное описание нагрузки, оказываемой потоком, позволяя службе легко определить требуемый объем ресурсов.

▪ Схема маркерного ведра предоставляет входные параметры для функции регулирования.

Спецификация трафика маркерного ведра включает два параметра:
▪ скорость пополнения маркеров R
▪ объем ведра B

Значение R определяет постоянную, установившуюся скорость передачи данных, то есть усредненную за относительно долгий период времени. Значение B определяет величину, на которую скорость передачи данных может превышать R в течение короткого интервала времени. В результате в течение любого интервала времени T количество переданных данных не может превышать RT+ B.

На рисунке показана данная схема и разъясняется использование термина **ведро**. Этим термином называют счетчик, указывающий количество байтов IP-данных, которые могут быть переданы в любой момент времени. Ведро заполняется байтовыми маркерами со скоростью R (то есть счетчик увеличивается на единицу R раз в секунду). Содержимое счетчика не может превышать определенного максимального значения, соответствующего объему ведра B. IP-пакеты прибывают и устанавливаются в очередь на обработку. IP-пакет может быть обработан, если в ведре есть достаточное количество маркеров (не менее размера IP-пакета). Если маркеров в ведре достаточно, пакет обрабатывается, а соответствующее количество маркеров вытекает из ведра. Если пакет прибывает, а в ведре нет достаточного количества маркеров, то это значит, что данный пакет превысил параметры спецификации трафика для данного потока. Что делать с таким пакетом, в документации архитектуры ISA не указывается. Обычно подобный пакет либо обслуживается по остаточному принципу (при наличии ресурсов), либо отбрасывается, либо особым образом помечается, чтобы его можно было отбросить позднее.

Средняя скорость передачи IP-данных, разрешенная маркерным ведром, равна R. Однако если возникает период простоя или период относительно низкой скорости поступления данных, ведро наполняется маркерами, что позволяет передать дополнительно до B байт сверх установленной скорости. Таким образом, объем ведра B представляет собой меру допустимой неравномерности потока данных.

Итак: 3 категории обслуживания: по остаточному принципу – все понятно, поэтому рассмотрим другие две.

Архитектура интегрированных служб

Гарантированное обслуживание:

- **Поддерживается гарантированный уровень пропускной способности или гарантированная скорость передачи данных.**
- **Определен верхний предел величины задержек пакетов в очередях.**
- **Нет потерь в очередях. То есть ни один пакет не теряется из-за переполнения буфера.**
  Пакеты могут теряться только из-за неисправностей в сети или изменений в маршрутах.

Контролируемая нагрузка:

- **Напоминает обслуживание с максимальными усилиями в условиях низкой загруженности сети.**
- **Не существует указанного верхнего предела задержек в очередях.**
  Однако гарантируется, что очень большой процент пакетов пройдет без задержек, значительно превышающих минимальную задержку прохождения данных по сети
- **Очень большой процент передаваемых пакетов будет успешно доставлен (то есть потери в очередях почти отсутствуют).**

6

Гарантированное обслуживание

Это наиболее требовательная служба в архитектуре ISA.

Ключевые характеристики гарантированного обслуживания:

▪ Поддерживается гарантированный уровень пропускной способности или гарантированная скорость передачи данных.

▪ Определен верхний предел величины задержек пакетов в очередях. Эта величина, сложенная с задержкой распространения, дает суммарную верхнюю границу задержки прохождения данных через сеть.

▪ Нет потерь в очередях. То есть ни один пакет не теряется из-за переполнения буфера. Пакеты могут теряться только из-за неисправностей в сети или изменений в маршрутах.

Этот класс обслуживания предназначен для приложений, которым необходима верхняя граница задержки, например для приложений, использующих входной буфер для накапливания и воспроизведения в режиме реального времени поступающих данных, не допускающих потери пакетов.

Контролируемая нагрузка

Ключевые характеристики обслуживания с контролируемой нагрузкой:

▪ Обслуживание с контролируемой нагрузкой очень напоминает обслуживание с максимальными усилиями в условиях низкой загруженности сети.

▪ Не существует указанного верхнего предела задержек в очередях. Однако гарантируется, что очень большой процент пакетов пройдет без задержек, значительно превышающих минимальную задержку прохождения данных по сети (то есть задержку, складывающуюся из времени распространения сигнала по сети и времени обработки пакетов на маршрутизаторе без потерь времени на ожидание в очередях).

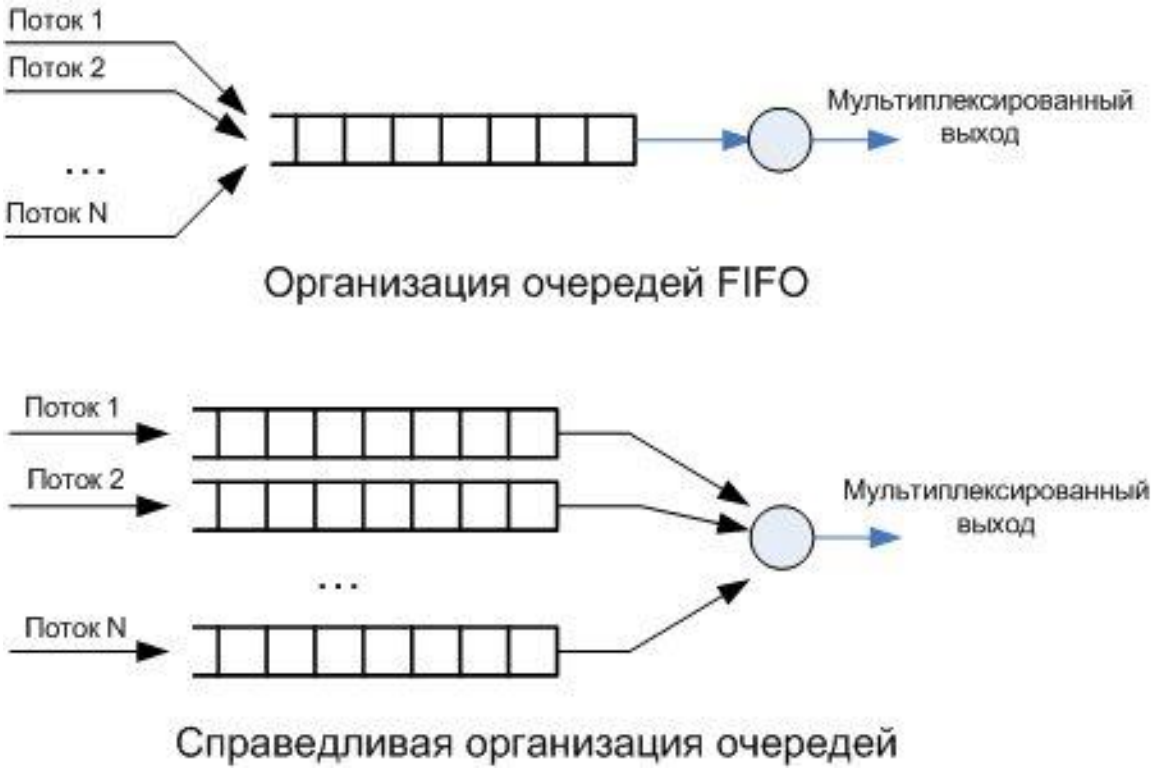
▪ Очень большой процент передаваемых пакетов будет успешно доставлен (то есть потери в очередях почти отсутствуют).

В сети, предоставляющей приложениям реального времени гарантии качества обслуживания, существует риск вытеснения из сети трафика, обслуживаемого по остаточному принципу. При обслуживании с контролируемой нагрузкой гарантируется, что сеть зарезервирует достаточно ресурсов для того, чтобы с точки зрения приложения сеть выглядела так, будто в ней не работают приложения реального времени, конкурирующие за ресурсы.

Обслуживание с контролируемой нагрузкой хорошо подходит для адаптивных приложений реального времени. Такие приложения не требуют заранее установленного верхнего ограничения на

задержки в сети. Например, приложение воспроизведения видеоданных можно адаптировать, отбросив кадр или слегка задержав выходной поток.

Дисциплина очередей



7

Дисциплина очередей

Традиционно маршрутизаторами на каждом выходном порту применяется дисциплина очередей FIFO и поддерживается отдельная очередь. Когда прибывает новый пакет и направляется к выходному порту, он помещается в конец очереди. Пока очередь не опустеет, маршрутизатор передает пакеты из очереди, выбирая из нее самый старый пакет.

Недостатки FIFO.

- Пакетам, принадлежащим потокам с более высоким приоритетом или потокам, в большей степени чувствительным к задержке, не предоставляется специального обслуживания. Если несколько пакетов из различных потоков готовы к передаче, они посылаются строго в порядке FIFO, то есть в том же порядке, в котором они поступили в очередь.
- Если несколько меньших по размеру пакетов стоят в очереди позади больших пакетов, тогда результат применения схемы FIFO будет выражаться в большей средней задержке пакета, чем в случае, если бы более короткие пакеты передавались прежде больших пакетов. В целом получается, что потоки из пакетов большего размера получают лучшее обслуживание.
- Более эгоистичное TCP-соединение может вытеснить более альтруистические TCP-соединения. Если возникает перегрузка и одно из TCP-соединений по какой-то причине не снизит скорость передачи данных, тогда другим TCP-соединениям придется пойти на большие уступки, чем в случае, если бы все TCP-соединения снизили свои скорости.

Справедливая организация очередей

Маршрутизатор обслуживает несколько очередей для каждого выходного порта. При этом можно поддерживать по одной очереди для каждого источника, как предложено в, или по одной очереди для каждого потока.

При справедливой организации очередей каждый входящий пакет помещается в соответствующую очередь. Очереди обслуживаются в циклическом порядке, то есть из каждой очереди поочередно выбирается один пакет. Пустые очереди пропускаются.

Такая схема является справедливой в том смысле, что каждому непустому потоку удается послать ровно один пакет за цикл. При такой схеме потоку нет смысла быть жадным. Единственное, что получит эгоистичный поток, это удлинение своей очереди и соответствующее увеличение задержки в доставке пакетов, при этом его поведение никак не скажется на других потоках.

Разделение процессора (Processor Sharing, PS)

Справедливая организация очередей с побитовым циклом (Bit-Round Fair Queuing, BRFQ)

$R(t)$ — количество циклов обслуживания с разделением процессора, которые произошли к моменту времени t, нормализованное относительно выходной скорости передачи данных;

$N(t)$ — количество непустых очередей в момент времени t;

P^{α}_{i} — время передачи для пакета i в очереди α, приведенное к выходной скорости передачи данных;

τ^{α}_{i} — время прибытия пакета i в очередь α;

S^{α}_{i} — значение $R(t)$ в момент начала передачи пакета i в очереди α;

F^{α}_{i} — значение $R(t)$ в момент завершения передачи пакета i в очереди α.

$$R'(t) = \frac{d}{dt} R(t) = \frac{1}{\max[1, N(t)]}$$

8

Разделение процессора

Серьезный недостаток схемы справедливой организации очередей заключается в том, что короткие пакеты получают менее качественное обслуживание. Потоки с большим средним размером пакета получают большую долю пропускной способности по сравнению с потоками с меньшим средним размером пакетов. Причина этого в том, что каждая очередь передает по одному пакету за цикл.

Этот недостаток можно преодолеть с помощью схемы справедливой организации очередей с побитовым циклом (Bit-Round Fair Queuing, BRFQ), в которой при планировании пакетов учитываются не только идентификаторы пакетов, но и длина пакетов.

Чтобы понять, как работает метод справедливой организации очередей с побитовым циклом, рассмотрим сначала идеальную политику, которая на практике не применяется. Сформируем несколько очередей, как в схеме справедливой организации очередей, но будем передавать в каждом цикле не

целый пакет, а всего один бит. Таким образом, более длинные пакеты не получат преимущества, и каждому источнику будет предоставлена равная пропускная способность.

В частности, если имеются N очередей и в каждой из этих очередей всегда есть пакет для передачи, тогда каждая очередь получает ровно 1/N доступной пропускной способности.

Такой идеальный побитовый подход известен также как разделение процессора (Processor Sharing, PS).

Чтобы дальнейшее обсуждение было понятнее, введем следующие обозначения:

- **R(t)** — количество циклов обслуживания с разделением процессора, которые произошли к моменту времени **t**, нормализованное относительно выходной скорости передачи данных;
- **N(t)** — количество непустых очередей в момент времени **t**;
- $\mathbf{P}^{\alpha}_{\mathbf{i}}$ — время передачи для пакета **i** в очереди **α**, приведенное к выходной скорости передачи данных;
- $\boldsymbol{\tau}^{\alpha}_{\mathbf{i}}$ — время прибытия пакета **i** в очередь **α**;
- $\mathbf{S}^{\alpha}_{\mathbf{i}}$ — значение **R(t)** в момент начала передачи пакета **i** в очереди **α**;
- $\mathbf{F}^{\alpha}_{\mathbf{i}}$ — значение **R{t)** в момент завершения передачи пакета **i** в очереди **α**.

Значение R(t) можно рассматривать как виртуальное время, соответствующее скорости обслуживания пакета, находящегося в начале очереди. Эквивалентное определение выглядит следующим образом:

$$R'(t) = \frac{d}{dt}R(t) = \frac{1}{\max[1, N(t)]}$$

Для примера рассмотрим три очереди, трафик которых описывается таблицей:

Разделение процессора

	Очередь **α**		Очередь **β**		Очередь **γ**
	Пакет 1	Пакет 2	Пакет 1	Пакет 2	Пакет 1
Реальное время прибытия $\boldsymbol{\tau}_{\mathbf{i}}$	**0**	**2**	**1**	**2**	**3**
Время передачи $\mathbf{P}_{\mathbf{i}}$	**3**	**1**	**1**	**4**	**3**
Виртуальное время начала $\mathbf{S}_{\mathbf{i}}$	**0**	**3**	**1**	**2**	**2**
Виртуальное время конца $\mathbf{F}_{\mathbf{i}}$	**3**	**4**	**2**	**6**	**5**

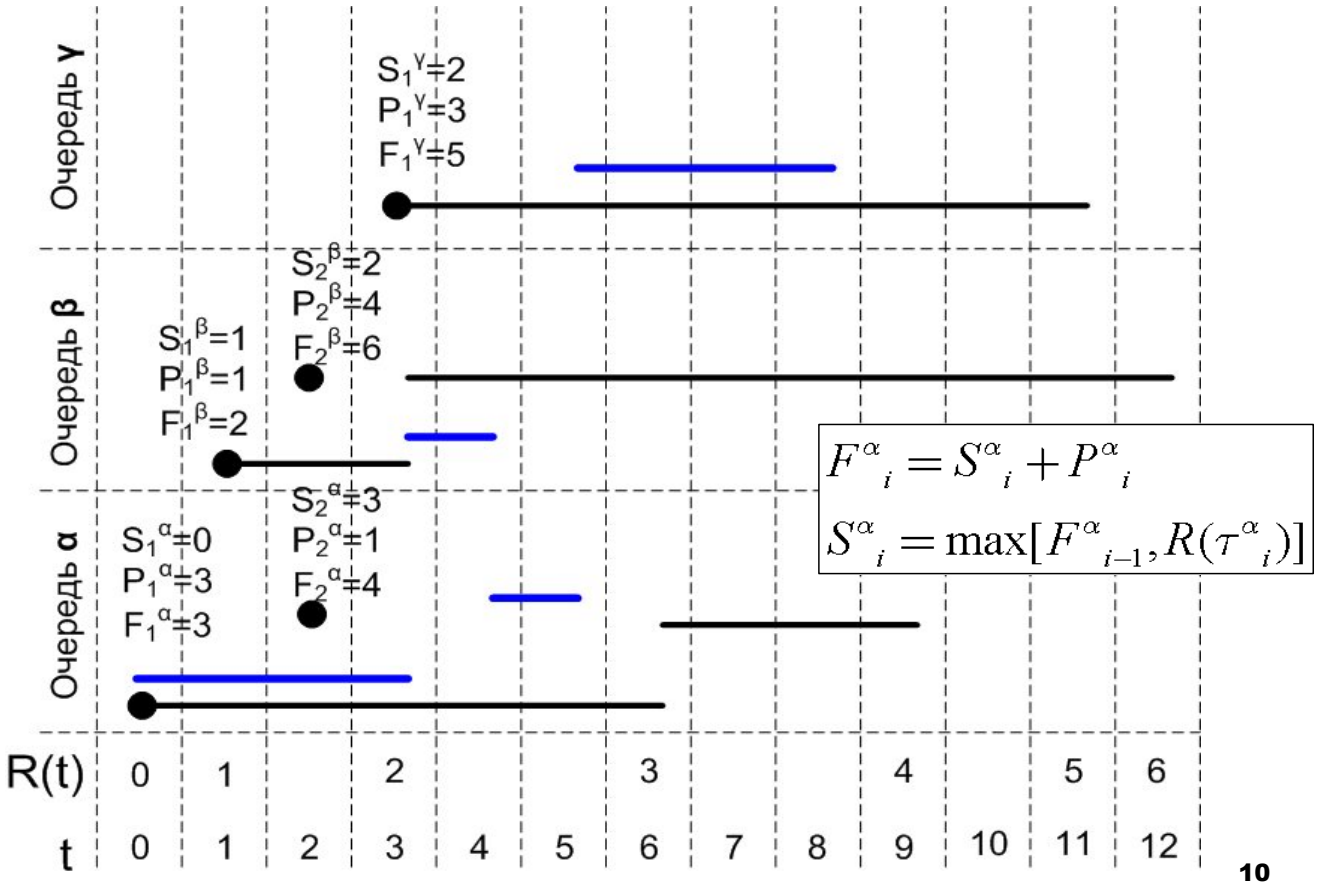
9

Таблица. Трафик трех очередей, обслуживаемый по схеме PS

	Очередь **α**		Очередь **β**		Очередь **γ**
	Пакет 1	Пакет 2	Пакет 1	Пакет 2	Пакет 1
Реальное время прибытия $\boldsymbol{\tau}_{\mathbf{i}}$	0	2	1	2	3

Время передачи $\mathbf{P_i}$	3	1	1	4	3
Виртуальное время начала $\mathbf{S_i}$	0	3	1	2	2
Виртуальное время конца $\mathbf{F_i}$	3	4	2	6	5

Разделение процессора



Значения τ_i, и $\mathbf{P_i}$, задаются; значения $\mathbf{S_i}$, и $\mathbf{F_i}$ определяются политикой разделения процессора. На рисунке черные линии иллюстрируют обслуживание трех очередей. Первый прибывший в очередь α пакет получает одну единицу обслуживания в реальном интервале времени [0, 1] с виртуальным временем R(l) = 1 к концу этого интервала.

В реальном интервале времени [1,3] биты передаются из двух очередей, в результате чего каждая очередь получает скорость обслуживания **R'**= 1/2 и виртуальное время растягивается в 2 раза. Аналогично, в течение интервала времени [3, 9] все три очереди сохраняют активность, поэтому каждая получает обслуживание со скоростью R'= 1/3 и виртуальное время растягивается в 2 раза.

Следующее рекуррентное соотношение показывает, как система разделения процессора развивается в виртуальном времени: это виртуальные времена завершения передачи пакета и начала передачи.

$$F^{\alpha}_{i} = S^{\alpha}_{i} + P^{\alpha}_{i}$$

$$S^{\alpha}_{i} = \max[F^{\alpha}_{i-1}, R(\tau^{\alpha}_{i})]$$

По этим соотношениям мы можем сосчитать виртуальное время окончания передачи пакета в момент прибытия пакета. Однако мы не можем вычислить реальное время окончания передачи пакета в момент его прибытия, потому оно зависит от прибытия других пакетов.

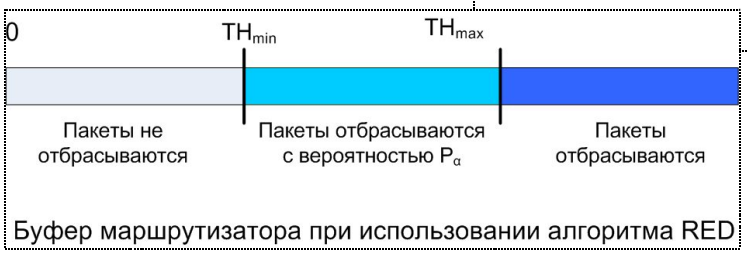
Случайное раннее обнаружение (Random Early Detection, RED)

Цели RED:

- **Предупреждение перегрузки**
- **Предотвращение глобальной синхронизации**
- **Предотвращение дискриминации источников неравномерного трафика**
- **Ограничение средней длины очередей**

Алгоритм

```
вычислить среднюю длину очереди avg
        если avg < THmin
                установить пакет в очередь
        иначе если THmin < avg < THmax
                вычислить вероятность Pa
                с вероятностью Pa
                        отбросить пакет
                иначе с вероятностью 1 - Pa
                        установить пакет в очередь
        иначе если avg > THmax
                        отбросить пакет
```



11

Случайное раннее обнаружение

Другой подход к борьбе с перегрузкой в объединенных сетях представляет собой упреждающее отбрасывание пакетов, т.е. маршрутизатор отбрасывает один или несколько входящих пакетов прежде, чем переполнится выходной буфер. Упреждающее отбрасывание пакетов может быть реализовано в одной или нескольких очередях.

Наиболее важный вариант схемы упреждающего отбрасывания пакетов называется случайное раннее обнаружение (Random Early Detection, RED). Сначала обсудим причины для применения схемы RED и цели ее разработки, после чего перейдем к деталям.

Мотивация

Когда в сети возникает перегрузка, буферы маршрутизаторов переполняются и маршрутизаторы начинают терять пакеты. Для TCP-трафика это является сигналом о необходимости перехода в т.н. «фазу затяжного пуска» для снижения нагрузки на сеть и устранения перегрузки. Это достаточно простая процедура, когда в начале передачи данных окно соединения устанавливается равным 1, а при получении каждого последующего подтверждения окно увеличивается на 1, пока не достигнет максимума.

У данного сценария есть два недостатка. Во-первых, потерянные пакеты необходимо передавать повторно, что увеличивает нагрузку на сеть и является причиной значительных задержек в TCP-потоках. Более серьезный феномен называется глобальной синхронизацией, при которой имеет место следующая ситуация. В случае резкого увеличения объемов трафика очереди переполняются, и множество пакетов теряется. Как правило, это затрагивает многие TCP-соединения, в результате они переходят в фазу затяжного пуска, суммарный объем сетевого трафика резко падает и в течение определенного периода сеть используется не в полной мере. Поскольку многие TCP-соединения

переходят в фазу затяжного пуска примерно в одно и то же время, они и выходят из этого режима также приблизительно одновременно, что вызывает очередное резкое увеличение объемов трафика в сети и новый цикл «изобилия и голодания».

Одно из решений заключается в использовании на каждом маршрутизаторе буферов большего размера для снижения вероятности потери пакетов. У такого подхода есть два недостатка. Во-первых, когда эти большие буферы заполняются, задержки во всех соединениях возрастают во много раз. Еще хуже ситуация в случае самоподобного трафика (инвариантный относительно изменения масштаба) – повторяющийся во всех масштабах времени. Тогда построить достаточно большие буферы невозможно в принципе. Большие всплески активности отправителей могут возникать друг за другом, поддерживая состояние перегрузки, и потребности в буферах при этом будут расти до бесконечности.

Лучшее решение состоит в том, чтобы предвидеть наступление перегрузки и предлагать то одному, то другому TCP-сообщению снизить скорость передачи данных. Затем определяется эффект этого замедления, после чего, в случае необходимости, замедляется другое соединение. Подобным образом, когда начинается перегрузка, торможение трафика осуществляется постепенно, с минимальным воздействием на TCP-соединения и без глобальной синхронизации. Именно такое решение реализовано в методе RED.

Цели разработки метода случайного раннего обнаружения

▪ Предупреждение перегрузки. Метод случайного раннего обнаружения предназначен не для реагирования на возникновение перегрузки, а для ее предотвращения.

▪ Предотвращение глобальной синхронизации. Когда маршрутизатор обнаруживает перегрузку, он должен решить, которому соединению (или соединениям) предложить снизить скорость передачи данных. В применяемом сегодня варианте этого метода данное уведомление является неявным и проявляется в потере пакетов. Обнаруживая перегрузку заранее и уведомляя о ней только те соединения, которые требуется, этот метод позволяет избежать глобальной синхронизации.

▪ Предотвращение дискриминации источников неравномерного трафика. С большой вероятностью перегрузка в сети начинается с поступления большого объема данных от одного или нескольких источников. Этот всплеск активности добавляется к текущей нагрузке на маршрутизатор. Если для отбрасывания выбираются только прибывающие пакеты, тогда велика вероятность того, что алгоритм отбрасывания пакетов будет направлен в основном против источников с непостоянной скоростью передачи данных.

▪ Ограничение средней длины очередей. Метод случайного раннего обнаружения должен уметь контролировать среднюю длину очередей и, следовательно, среднюю задержку.

Алгоритм RED

```
вычислить среднюю длину очереди avg
если avg < THmin
установить пакет в очередь
иначе если THmin < avg < THmax
вычислить вероятность Pa
с вероятностью Pa
отбросить пакет
иначе с вероятностью 1 - Pa
установить пакет в очередь
иначе если avg > THmax
отбросить пакет
```

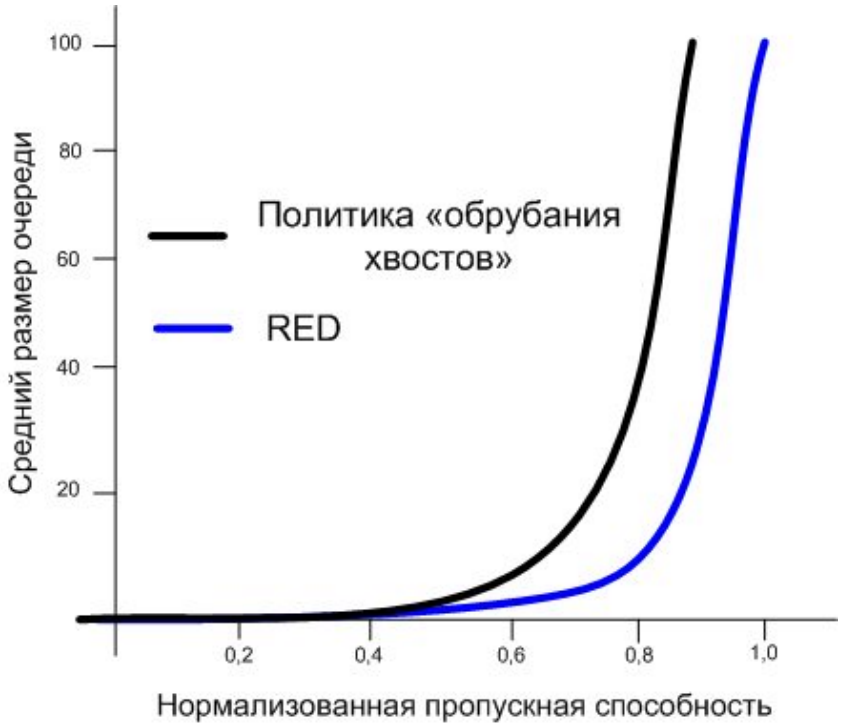
Каждый раз, когда новый пакет поступает в выходную очередь с дисциплиной FIFO, этот алгоритм выполняет две функции. Первый шаг заключается в вычислении средней длины очереди avg. Средняя длина очереди сравнивается с двумя уровнями. Если средняя длина очереди avg меньше нижнего предела THmin, то перегрузка считается минимальной или отсутствующей и пакет помеща-

ется в очередь. Если значение avg больше верхнего предела THmax, то перегрузка считается серьезной и пакет отбрасывается. Если значение avg находится между двумя предельными значениями, тогда мы попадаем в область перегрузки. В этой области вычисляется вероятность выбрасывания пакета P_a, зависящая от точного значения средней длины очереди avg и увеличивающаяся при приближении

значения avg к верхнему пределу. Когда средняя длина очереди находится в этой области, пакет отбрасывается с вероятностью P,, и ставится в очередь с вероятностью 1 - Pa.

По сути, первая часть алгоритма (вычисление средней длины очереди) определяет допустимый уровень неравномерности трафика, а вторая часть алгоритма — частоту отбрасывания пакетов при данном уровне перегрузки.

Случайное раннее обнаружение



12

На рисунке показан результат эмуляции, в которой алгоритм RED сравнивается с политикой обрубания хвостов (drop-tail policy), когда прибывающий пакет просто отбрасывается, если в очереди нет свободного места. При высоких уровнях перегрузки алгоритм RED заметно превосходит политику обрубания хвостов, обеспечивая более высокую пропускную способность.

Дифференциальные службы (Differentated Services, DS)
RFC2475

- **Для предоставления различных классов обслуживания IP-пакеты помечаются соответствующим образом (используется поле типа службы протокола IPv4 или поле класса трафика IPv6.**
  Таким образом, не требуется никаких изменений в протоколе IP.

- **Перед использованием дифференцированной службы между поставщиком услуг и пользователем устанавливается соглашение об уровне обслуживания.**

- **Дифференцированные службы предоставляют встроенный механизм агрегирования. Весь трафик с одинаковым полем DS обрабатывается сетевой службой одинаково.**

- **Дифференцированные службы реализуются на отдельных маршрутизаторах при установке пакетов в очередь и переправке их дальше в сеть на основе значения поля DS.**

13

Дифференцированные службы

Хотя архитектура ISA в целом и протокол резервирования ресурсов RSVP в частности, являются полезными инструментами, их довольно трудно реализовать. Более того для больших объемов трафика они плохо масштабируются из-за большого количества управляющих сигналов, а также в связи с необходимостью поддерживать в маршрутизаторах информацию о состоянии.

Архитектура дифференцированных служб (Differentiated Services, DS) – RFC 2475, предназначена для предоставления простого и легкого в реализации механизма поддержания сетевых служб, различающихся по производительности.

Ключевые характеристики дифференцированных служб:

- Для предоставления различных классов обслуживания IP-пакеты помечаются соответствующим образом, для этого используется поле типа службы протокола IPv4 или поле класса трафика IPv6. Таким образом, не требуется никаких изменений в протоколе IP.
- Перед использованием дифференцированной службы между поставщиком услуг и пользователем устанавливается соглашение об уровне обслуживания. Это позволяет избежать необходимости встраивания дифференцированных служб в приложения. Т.е., не нужно изменять существующие приложения.
- Дифференцированные службы предоставляют встроенный механизм агрегирования (многократного использования). Весь трафик с одинаковым полем DS обрабатывается сетевой службой одинаково. Например, несколько голосовых соединений не обрабатываются индивидуально, а обслуживаются вместе. Это обеспечивает хорошую масштабируемость для сетей большего размера и большей нагрузки.
- Дифференцированные службы реализуются на отдельных маршрутизаторах при установке пакетов в очередь и переправке их дальше в сеть на основе значения поля DS. Маршрутизаторы

индивидуально обрабатывают каждый пакет, и у них нет необходимости сохранять информацию о состоянии потоков пакетов.

Дифференциальные службы

Параметры производительности:

- **Ожидаемая пропускная способность, вероятность отбрасывания пакета, задержка.**
- **Ограничения на входные и выходные точки, в которых предоставляется услуга.**
- **Профили трафика, которые должны выдерживаться для запрашиваемой услуги, например параметры маркерного ведра.**
- **Размещение трафика, поставляемого сверх указанного профиля.**

Примеры предоставляемых услуг:

- **Трафик, предлагаемый на уровне обслуживания класса A, будет доставлен с низкой задержкой.**
- **Трафик, предлагаемый на уровне обслуживания класса B, будет доставлен с низкими потерями.**
- **Задержки для 90 % трафика, предлагаемого на уровне обслуживания класса C, не превысят 50 мс.**
- **Получателю будет доставлено 95 % трафика, предлагаемого на уровне обслуживания класса D.**
- **Трафику, предлагаемому на уровне обслуживания класса E, будет предоставлена в два раза большая пропускная способность, чем трафику, предлагаемому на уровне обслуживания класса F.**
- **Трафик с очередностью отбрасывания X доставляется с большей вероятностью, чем трафик с очередностью отбрасывания Y.**

14

Службы

Услуги, предоставляемые в системе дифференцированных служб, определяются в соглашении об уровне обслуживания, это договор об обслуживании между пользователем и поставщиком услуг.

Как только соглашение об уровне обслуживания достигнуто, пользователь начинает поставлять пакеты, в которых в поле DS указывается класс пакета. Поставщик услуг должен гарантировать, что пользователь получит по меньшей мере договорный уровень качества обслуживания для каждого класса пакетов.

Если пользователь поставляет пакеты, направляемые получателям в том же домене дифференцированных служб, тогда от домена ожидается предоставление указанных в договоре услуг. Если же получатель находится за пределами DS-домена пользователя, тогда домен попытается переправить пакеты через другие домены, запросив обслуживание, лучше всего соответствующее запрашиваемому уровню.

Параметры производительности, которые могут быть включены в соглашение об уровне обслуживания:

- Ожидаемая пропускная способность, вероятность отбрасывания пакета, задержка.
- Ограничения на входные и выходные точки, в которых предоставляется услуга, указывающие на область ее применения.
- Профили трафика, которые должны выдерживаться для запрашиваемой услуги, например параметры маркерного ведра.
- Размещение трафика, поставляемого сверх указанного профиля.

Примеры предоставляемых услуг:

- Трафик, предлагаемый на уровне обслуживания класса A, будет доставлен с низкой задержкой.
- Трафик, предлагаемый на уровне обслуживания класса B, будет доставлен с низкими потерями.
- Задержки для 90 % трафика, предлагаемого на уровне обслуживания класса C, не превысят 50 мс.
- Получателю будет доставлено 95 % трафика, предлагаемого на уровне обслуживания класса D.
- Трафику, предлагаемому на уровне обслуживания класса E, будет предоставлена в два раза большая пропускная способность, чем трафику, предлагаемому на уровне обслуживания класса F.
- Трафик с очередностью отбрасывания X доставляется с большей вероятностью, чем трафик с очередностью отбрасывания Y.

Первые два примера являются качественными и имеют смысл только при сравнении с другим трафиком, например трафиком по умолчанию, получающим обслуживание с максимальными усилиями (то есть по остаточному принципу). Следующие два примера являются количественными и предоставляют определенные гарантии. Последние два примера представляют собой сочетание качественных и количественных характеристик.

Дифференциальные службы

Поле DS (RFC 2474)

Подполе очередности:

101 — критически важный пакет;
010 — пакет требует немедленной реакции;
000 — регулярный пакет.

Подполе типа службы:

1000 — минимизировать задержку;
0100 — максимизировать пропускную способность;
0010 — максимизировать надежность;
0001 — минимизировать денежную стоимость;
0000 — обычное обслуживание.

15

Поле DS

Пакеты маркируются при помощи поля DS, в поле типа службы заголовка IPv4 или в поле класса трафика заголовка IPv6. В документе RFC 2474 определяется следующий формат поля DS. Первые 6 бит образуют DS-код, а последние 2 бита временно не используются. На рис. 17.10 показано поле DS и более ранняя интерпретация поля типа службы протокола IPv4. Ниже перечислены значения подполей очередности и типа службы для протокола IPv4.

Подполе очередности:

- 101 — критически важный пакет;
- 010 — пакет требует немедленной реакции;
- 000 — регулярный пакет.

Подполе типа службы:

- 1000 — минимизировать задержку;
- 0100 — максимизировать пропускную способность;
- 0010 — максимизировать надежность;
- 0001 — минимизировать денежную стоимость;
- 0000 — обычное обслуживание.

- С помощью 6-битового кода, в принципе, можно определить 64 различных класса трафика.

Код 000000 объявлен классом пакетов по умолчанию. Этот класс обслуживается маршрутизаторами с максимальными усилиями (по остаточному принципу). Такие пакеты переправляются в том же порядке, в котором они были получены, как только линия связи становится доступной. Пакеты более высоких классов получают преимущество по сравнению с пакетами класса 000000.

Конфигурирование и работа дифференцированных служб

Маршрутизаторы в домене дифференцированных служб представляют собой либо пограничные узлы, либо внутренние узлы.

Внутренними узлами реализуются простые механизмы обработки пакетов на основе значений их полей DS. Эти механизмы включают дисциплину очередей, обслуживание в зависимости от значений поля DS, а также правила отбрасывания пакетов в случае наполнения буфера.

В спецификации дифференцированных служб обработка пакетов на маршрутизаторе называется поведением на ретрансляционном участке (Per-Hop Behavior, PHB).

На пограничных узлах также реализованы эти механизмы, но кроме них они содержат более сложные механизмы согласования, необходимые для предоставления требуемой услуги. Таким образом, внутренние маршрутизаторы обладают минимальной функциональностью, а самые сложные механизмы реализованы на пограничных узлах.

Функция согласования трафика состоит из пяти элементов:

- **Классификатор**. Разделяет поставляемые пакеты на различные классы. Это основа предоставления дифференцированных услуг. Классификатор может разделять трафик только на основе значения кода DS или на основе нескольких полей заголовка пакета, или даже по содержимому полезной нагрузки пакета.
- **Измеритель**. Измеряет предоставляемый трафик на предмет соответствия профилю. Измеритель определяет, находится ли данный класс потока пакетов в установленных пределах или превосходит уровень обслуживания, гарантированный для этого класса.
- **Маркировщик**. Управляет трафиком, при необходимости заново маркируя пакеты различными кодами. Например, если для определенного класса обслуживания гарантируется некоторая пропускная способность, любые пакеты этого класса, превышающие пропускную способность в течение определенного интервала времени, могут быть маркированы заново для обслуживания по остаточному принципу. Кроме того, изменение маркировки может потребоваться на границе между двумя доменами дифференцированных служб. Например, данный класс трафика должен получать наивысший поддерживаемый приоритет, а максимальный приоритет в одном домене — 3, а в другом — 7. В этом случае при переходе из первого домена во второй пакеты с приоритетом 3 должны маркироваться заново как пакеты с приоритетом 7.
- **Формирователь**. Управляет трафиком, задерживая при необходимости пакеты таким образом, чтобы поток пакетов данного класса не превышал скорости трафика, указанного в профиле данного класса.
- **Отбрасыватель**. Отбрасывает пакеты, когда скорость пакетов данного класса превышает скорость, указанную в профиле данного класса.

После того как поток классифицирован, следует измерить, сколько ресурсов он потребляет. Функция измерения определяет суммарный объем пакетов за определенный период времени, чтобы выяснить, соответствует ли поток соглашению о трафике.

Если трафик превышает определенный профиль, могут предприниматься различные действия. У отдельных пакетов, превышающих профиль, может быть изменена маркировка в сторону понижения класса обслуживания. Формирователь трафика может поглотить всплеск трафика с помощью своего буфера и распределить принятые пакеты на более длительный период времени. Отбрасыватель может отбросить пакеты, если используемый для регулирования скорости буфер заполнен.

Дифференциальные службы

В спецификации дифференцированных служб обработка пакетов на маршрутизаторе называется поведением на ретрансляционном участке (Per-Hop Behavior, PHB).

Два стандарта, описывающих типы поведения на ретрансляционном участ-ке:
- срочное продвижение данных (RFC 2598)
- гарантированное продвижение данных (RFC 2597)

Срочное продвижение данных (Expedited Forwarding, EF)

Премиальная услуга – сквозное обслуживание с низкими потерями, низкой задержкой, низкими флуктуациями и гарантированной пропускной способностью.

Агрегат трафика – поток пакетов, ассоциированный с обслуживанием конкретного пользователя.

Премиальная услуга подразумевает выполнение двух условий:

1. **Конфигурирование узлов таким образом, чтобы у агрегата трафика была четко определенная минималь-ная скорость отправления.**
2. **Согласование агрегата таким образом, чтобы скорость его прибытия на любой узел всегда была ниже минимальной скорости отправления, сконфигурированной для данного узла.**

16

Поведение на ретрансляционном участке

Два стандарта, описывающих типы поведения на ретрансляционном участке:

- срочное продвижение данных (RFC 2598)
- гарантированное продвижение данных (RFC 2597)

Срочное продвижение данных

Expedited Forwarding, EF - механизм, который может использоваться для предоставления так называемой премиальной услуги. Премиальная услуга – сквозное обслуживание с низкими потерями, низкой задержкой, низкими флуктуациями и гарантированной пропускной способностью. По существу, с точки зрения конечных потребителей это выглядит как соединение «точка — точка» или выделенная линия.

В объединенной сети или сети с коммутацией пакетов предоставить премиальную услугу чрезвычайно сложно. По своей природе объединенная сеть включает очереди на каждом узле или маршрутизаторе, когда пакеты помещаются в буфер, ожидая, пока освободится выходная линия коллективного доступа. Именно от поведения этих очередей зависят такие параметры соединения, как потери, задержки и флуктуация. Таким образом, при управлении трафиком, которому предоставляется премиальная услуга, должны предприниматься специальные меры, гарантирующие, что ожидание пакетов в очередях не приведет к превышению заданных пределов значениями задержки, потерь и флуктуации.

Агрегат трафика – поток пакетов, ассоциированный с обслуживанием конкретного пользователя.

Премиальная услуга подразумевает выполнение двух условий:

- Конфигурирование узлов таким образом, чтобы у агрегата трафика была четко определенная минимальная скорость отправления.
- Согласование агрегата таким образом, чтобы скорость его прибытия на любой узел всегда была ниже минимальной скорости отправления, сконфигурированной для данного узла.

Срочное продвижение данных позволяет выполнить первое из этих условий, тогда как второе условие выполняется сетевыми пограничными формирователями.

Дифференциальные службы

Гарантированное продвижение данных (Assured Forwarding, EF)

Схема явного выделения ресурсов:

1. **Пользователям предоставляется выбор из нескольких классов обслуживания для их трафика.**
2. **Поступающий от пользователя трафик наблюдается на пограничном узле. Каждый пакет трафика маркируется в зависимости от того, соответствует он профилю трафика или нет.**
3. **В сети не производится разграничения трафика разных пользователей и разных классов обслуживания. Вместо этого весь трафик обрабатывается как единое множество пакетов, отличающихся друг от друга только маркировкой.**
4. **В случае возникновения перегрузки внутренние узлы задействуют схему отбрасывания пакетов, в первую очередь отбрасывающую пакеты, маркированные как не соответствующие профилю трафика.**
5. **Различные пользователи получат разные уровни обслуживания, так как в очередях на обслуживание у них будет разное количество пакетов, маркированных как соответствующие профилю трафика.**

Схема гарантированного продвижения данных:

1. **Определяются четыре класса гарантированного продвижения данных. В каждом классе пакеты маркируются клиентом или поставщиком услуг одним из трех значений очередности отбрасывания. При возникновении перегрузки по маркировке очередности отбрасывания определяется относительная важность пакета в пределах класса гарантированного продвижения данных.**
2. **Перегруженный DS-узел пытается защитить пакеты с более низким значением очередности отбрасывания от потерь, отбрасывая в первую очередь пакеты с более высоким значением очередности отбрасывания.**

17

Гарантированное продвижение данных

Assured Forwarding, AF – призвано предоставлять обслуживание, лучшее, чем обслуживание по остаточному принципу, но не требующее резервирования ресурсов в объединенной сети, а также не требующее детального разграничения потоков разных пользователей. В основе гарантированного продвижения данных лежит понятие явного выделения ресурсов. Гарантированное продвижение данных сложнее, чем явное выделение ресурсов, но будет полезным сначала рассмотреть ключевые элементы схемы явного выделения ресурсов:

1. Пользователям предоставляется выбор из нескольких классов обслуживания для их трафика.
2. Поступающий от пользователя трафик наблюдается на пограничном узле. Каждый пакет трафика маркируется в зависимости от того, соответствует он профилю трафика или нет.
3. В сети не производится разграничения трафика разных пользователей и разных классов обслуживания. Вместо этого весь трафик обрабатывается как единое множество пакетов, отличающихся друг от друга только маркировкой.

4. В случае возникновения перегрузки внутренние узлы задействуют схему отбрасывания пакетов, в первую очередь отбрасывающую пакеты, маркированные как не соответствующие профилю трафика.

5. Различные пользователи получат разные уровни обслуживания, так как в очередях на обслуживание у них будет разное количество пакетов, маркированных как соответствующие профилю трафика.

Определенная в RFC 2597 схема гарантированного продвижения данных расширяет предыдущий подход в нескольких направлениях:

- Определяются четыре класса гарантированного продвижения данных, что позволяет указать четыре разных профиля трафика. Пользователь может выбрать один или несколько из этих классов для удовлетворения своих требований.
- В каждом классе пакеты маркируются клиентом или поставщиком услуг одним из трех значений очередности отбрасывания. При возникновении перегрузки по маркировке очередности отбрасывания определяется относительная важность пакета в пределах класса гарантированного продвижения данных. Перегруженный DS-узел пытается защитить пакеты с более низким значением очередности отбрасывания от потерь, отбрасывая в первую очередь пакеты с более высоким значением очередности отбрасывания.

Такой подход еще проще реализовать, чем любую разновидность схемы резервирования ресурсов. На внутреннем DS-узле трафик четырех различных классов может обрабатываться раздельно, при этом четырем классам будет выделяться разный объем ресурсов (буферное пространство, скорость передачи данных).

Дифференциальные службы

Рекомендованные DS-коды для гарантированного продвижения данных (AF) в качестве типа поведения на ретрансляционном участке.

Селектор класса:
100 – класс 4 (наилучшее обслуживание);
011 – класс 3;
010 – класс 2;
001 – класс 1.

Очередность отбрасывания:
010 — низкая (наиболее важные пакеты);
100 — средняя;
110 — высокая (наименее важные пакеты).

Поле DS

18

Рекомендованные DS-коды для гарантированного продвижения данных (AF) в качестве типа поведения на ретрансляционном участке.

Селектор класса:
- 100 – класс 4 (наилучшее обслуживание);
- 011 – класс 3;
- 010 – класс 2;
- 001 – класс 1.

Очередность отбрасывания:
- 010 — низкая (наиболее важные пакеты);
- 100 — средняя;
- 110 — высокая (наименее важные пакеты).

Лекция 8. Теоретические основы сжатия данных.

Прежде чем начать обсуждение проблем сжатия данных, освоим теоретический фундамент — теорию информации. Основы теории информации были заложены Клодом Шенноном (Claude Shannon) в процессе исследования пропускной способности информационного канала. Теория информации получила самые разнообразные применения. Для настоящего обсуждения важно то, что теория информации определяет предел, до которого для заданного потока данных можно сжимать информацию без потерь.

Информация и энтропия

Основу теории информации составляют две математические концепции, названия которых могут ввести в заблуждение: информация и энтропия. Как правило, под *информацией* (information) подразумевается нечто, относящееся к смыслу, а *энтропия* (entropy) — термин из второго закона термодинамики. В теории информации информация имеет отношение к снижению неосведомленности о некоем событии, а энтропией называется усреднение информационных значений, подчиняющееся тем же математическим законам, что и термодинамическая энтропия. Рассмотрим это новое определение информации на примере. Представим инвестора, которому требуется *информация* (совет) о состоянии определенных ценных бумаг. Этот инвестор советуется с брокером, обладающим специальной *информацией* (знанием) в данной области. Брекер *информирует* (сообщает) инвестора, что сегодня утром нагрянул федеральный инспектор, искавший *информацию* (свидетельства) о возможном мошенничестве, в котором замешана корпорация, выпустившая именно эти акции. В ответ на эту *информацию* (данные) инвестор решает продать свои акции, о чем и *информирует* (уведомляет) брокера. Другими словами, будучи *неуверенным* в вопросе о том, как распорядиться своим портфелем ценных бумаг, клиент консультируется с кем-то более *уверенным* в данном вопросе. Брокер уменьшает *неуверенность* клиента в этой области, рассказав ему о визите федерального инспектора, который пришел, чтобы разрешить собственную профессиональную *неуверенность.* Кульминацией возрастающей *уверенности* клиента о состоянии своих ценных бумаг становится устранение *неуверенности* брокера о намерении клиента продать эти ценные бумаги. Хотя термин *информация* может означать уведомление, знание или просто данные, в каждом случае получение информации эквивалентно уменьшению неуверенности. Таким образом, информация означает положительную разность между двумя уровнями неуверенности.

Информация

Чтобы рассматривать понятие информации с точки зрения математики, нам нужна определенная величина, годящаяся для измерения количества информации. Впервые эту проблему сформулировал и решил Хартли (Hartley) в 1928 г., изучая телеграфную связь. Хартли заметил, что если вероятность того, что некоторое событие произойдет, высока (близка к 1), неуверенность в том, что это событие произойдет, невелика. Если впоследствии мы узнаем, что это событие произошло, то количество полученной нами информации будет невелико. Таким образом, внушающей доверие мерой информации может служить величина, обратная вероятности некоего события, — *1/p.* Например, если происходит событие, вероятность которого равна 0,25, то мы получаем больше информации, чем если произойдет событие, вероятность которого равна 0,5. Если мера информации равна 1/p, тогда первое событие обладает информацией 4 (1/0,25), а количество информации о втором событии равно 2 (1/0,5). Но в использованиии этой меры количества информации есть трудности:

- Эта единица измерения, кажется, «не работает» с последовательностями событий. Рассмотрим двоичный источник, генерирующий поток из единиц и нулей с равной вероятностью значения каждого бита. Таким образом, каждый бит несет количество информации, равное 2 (1/0,5). Но если бит b_1 несет в себе 2

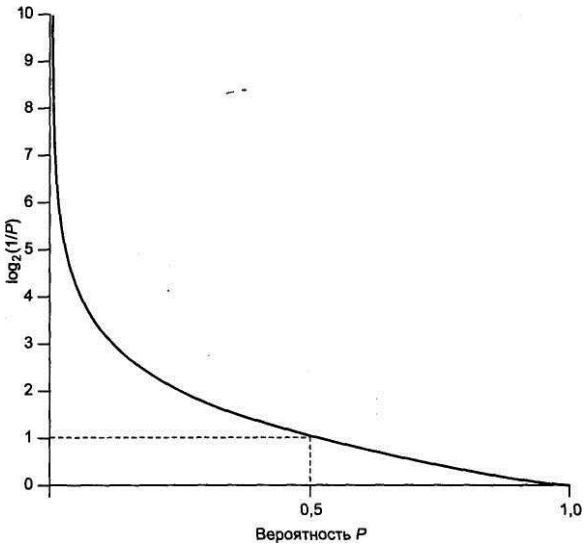
единицы информации, то сколько информации содержится в строке из двух битов b_1 b_2? Эта строка может содержать одну из четырех возможных комбинаций битов, причем каждое из четырех значений строка может принимать с вероятностью 0,25. Таким образом, при использовании в качестве меры количества информации величину 1/p два бита будут содержать четыре единицы информации. Продолжая данные рассуждения, количество информации, содержащееся в трех битах (b_1,b_2,b_3), равно восьми. Это означает, что первые два бита, b_1 и b_2, несут по две единицы информации, а третий бит (b_3) добавляет сразу четыре единицы информации. Следующий бит (b_4) добавит к последовательности уже восемь единиц информации и т. д. Это не кажется разумным.

Рассмотрим событие, приводящее к изменению двух или более независимых переменных. Примером такого события может быть сигнал, кодируемый при помощи так называемого метода фазовой манипуляции, для которого используется четыре возможных фазы и две амплитуды. Один элемент такого сигнала содержит две единицы информации для амплитуды и четыре для фазы, итого шесть единиц. В то же время, каждый элемент сигнала может иметь восемь возможных значений, и поэтому должен соответствовать восьми единицам информации, если пользоваться принятой нами мерой. Хартли разрешил данную проблему, прологарифмировав данную величину, то есть он предложил в качестве меры количества информации события x функцию $\log(1/P(x))$, где $P(x)$ означает вероятность события x. Формально можно написать:

$$I(x) = \log (1/P(x)) = - \log P(x)$$

Эта мера количества информации «работает» и позволяет получить множество полезных результатов.

Основание логарифма может быть взято произвольно, но удобнее всего логарифмировать по основанию 2, в этом случае единица информации называется битом.



На рисунке показан график зависимости количества информации единичного события от вероятности этого события. Когда вероятность события приближается к единице (событие происходит почти наверняка), количество информации, содержащееся в данном событии, приближается к нулю. И наоборот, когда вероятность события приближается к нулю (событие практически невероятно), количество информации, содержащееся в данном событии, приближается к бесконечности.

Энтропия

Другая важная концепция теории информации — *энтропия.* Эта концепция была предложена в 1948 г. основателем теории информации Шенноном (Shannon). Шеннон определил энтропию как среднее количество информации, получаемое от значения случайной переменной. Предположим, что имеется случайная переменная X, способная принимать значения x_1 x_2,..., x_N, и что соответствующие вероятности каждого исхода равны $P(x_1)$, $P(x_2)$,..., $P(x_N)$. В последовательности из K переменных X результат X_j в среднем будет выбран $KP(X_j)$ раз. Таким образом, среднее количество информации, получаемое от K событий, равно (будем использовать обозначение P_j для $P(x_j)$):

$$KP_i \log (1/P1) + ... + KP_N \log (1/P_N).$$

Если разделить это выражение на *K,* то мы получим среднее количество информации для одного результата случайной переменной, называемое энтропией *X* и обозначаемое как *H(X):*

$$H(X) = \sum_{j=1}^{N} P_j \log(1/P_j) = -\sum_{j=1}^{N} P_j \log(P_j).$$

Функция *H* часто выражается как перечень вероятностей возможных результатов: $H(P_1, P_2, \ldots, P_N)$.

Для примера рассмотрим случайную переменную *X,* принимающую два возможных значения с соответствующими вероятностями *P* и 1 - *P.* В этом случае ассоциированная с *X* энтропия будет равна:

$$H(P, 1 - P) = - P \log(P) - (1 - P)\log(1 - P).$$

На рисунке ниже, показан график *H(X)* для данного случая как функция от *P.* По этому графику можно отметить несколько важных особенностей энтропии. Во-первых, если одно из двух событий является достоверным (P= 1 или P=0), тогда энтропия равна нулю. Одно из двух событий должно произойти, и никакой информации о том, что одно из них произошло, не содержится. Во-вторых, максимального значения функция *H(X)* достигает, когда два результата равновероятны. Это также обоснованно: когда два результата равновероятны, неуверенность в результате максимальна. Этот результат можно обобщить для случайной переменной с *N* результатами. Энтропия случайной переменной будет максимальна, когда все результаты равновероятны:

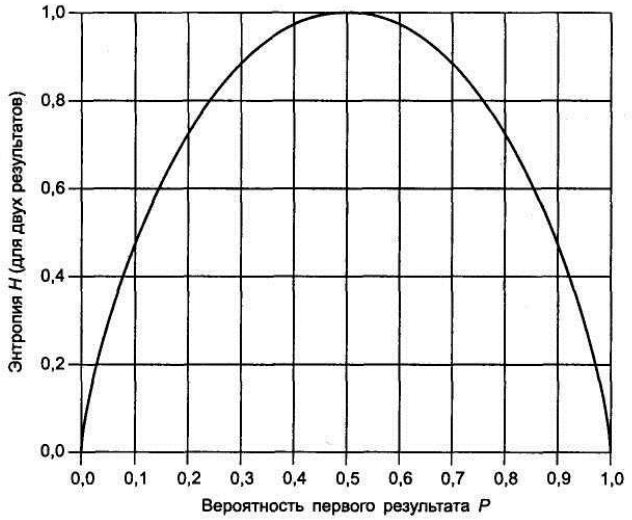
$$\max H(P_1, P_2, \ldots, P_N) = H(1/N, 1/N, \ldots, 1/N).$$

Например,

$$H(1/3, 1/3, 1/3) = 1/3 \log 3 + 1/3 \log 3 + 1/3 \log 3 \approx 1{,}585,$$

тогда как

$$H(1/2, 1/3, 1/6) = 1/2 \log 2 + 1/3 \log 3 + 1/6 \log 6 \approx 0{,}5 + 0{,}528 + 0{,}43 = 1{,}458.$$



Кодирование

Одно из важных приложений концепции энтропии заключается в том, что эта концепция помогает понять принциды работы алгоритмов сжатия данных. Энтропия случайной переменной или источника сообщений определяет количество битов, требуемых для представления результатов случайной переменной или альтернативных сообщений без потери информации. Таким образом, при разработке алгоритма сжатия данных энтропия представляет собой меру максимально возможного.

Код Хаффмана

Рассмотрим случайную переменную *X,* принимающую одно из восьми возможных значений $(x_1, x_2, \ldots, x_8)$ со следующими вероятностями:

$P_1 = 0{,}512$, $P_5 = 0{,}032$,
$P_2 = 0{,}128$, $P_6 = 0{,}032$,
$P_3 = 0{,}128$, $P_7 = 0{,}032$,
$P_4 = 0{,}128$, $P_8 = 0{,}008$.

Здесь P_i представляет собой вероятность выпадения результата x_i. Предположим, что сообщение состоит из реализаций случайной переменной *X*, и мы хотели бы закодировать сообщение в двоичном виде. Один очевидный вариант решения заключается в том, чтобы использовать 3-битовый код фиксированной длины, в котором каждое из восьми возможных значений случайной переменной *X* кодируется одним 3-битовым числом. Лучшая стратегия заключается в использовании кода переменной длины, в котором более длинные кодовые слова назначаются менее вероятным значениям *X*, а более вероятные значения *X* кодируются короткими кодовыми словами. Такой технический прием применяется в азбуке Морзе и коде Хаффмана. Предположим, что сообщения должны передаваться при помощи алфавита из N символов. Каждый символ должен уникальным образом кодироваться двоичной последовательностью. Нас интересует способ построения оптимального кода, то есть кода, дающего в результате минимальную среднюю длину кодируемого сообщения. Важно отметить, что мы не ищем минимальную длину кода для какого-либо конкретного сообщения или для всех сообщений (последнее найти невозможно), но минимальную длину кода, усредненную по всем возможным сообщениям.

Другой способ взглянуть на данные требования состоит в том, что мы получаем сообщение, уже закодированное путем назначения символам двоичных слов фиксированной длины. Таким образом, если символов 8, каждый символ кодируется тремя битами. Если число символов в алфавите от 9 до 16, то каждый символ кодируется четырьмя битами и т. д. Такое кодирование не является оптимальным, если символы встречаются в сообщениях с разной вероятностью. В этом случае требование может быть сформулировано следующим образом. Требуется разработать оптимальную схему кодирования с использованием кода переменной длины, позволяющую получить кодированное сообщение минимальной средней длины.

Рассмотрим двоичный код с кодовыми длинами $L_1, L_2 \ldots L_N$, поставленный в соответствие алфавиту из *N* символов с вероятностями $P_1, P_2, \ldots, P_N$. Для удобства предположим, что символы организованы в порядке убывания вероятности ($P_1 >= P_2 >= \ldots >= P_N$). Можно показать, что оптимальный код должен удовлетворять следующим требованиям:

- Никакие два разных сообщения не должны состоять из одинаковых после довательностей битов.
- Никакое кодовое слово не должно совпадать с префиксом другого кодового слова.
- Символы, встречающиеся с большей вероятностью, должны кодироваться более короткими кодовыми словами. То есть $L_1 <= L_2 <= \ldots <= L_N$.
- Два кодовых слова для символов с наименьшей вероятностью должны иметь одинаковую длину $(L_N = L_{N-1})$ и различаться только последней цифрой.

Первое требование гарантирует уникальность кодирования любого сообщения. Второе требование определяет так называемый *мгновенный* код. Это требование не является строго обязательным, но оно требуется, чтобы гарантировать, что сообщение может быть декодировано шаг за шагом. При продвижении слева направо, как только последовательность битов совпадает с данным кодовым словом, декодирующий алгоритм может произвести соответствующее назначение. Вот пример кода, нарушающего первые два требования:

x_1	0
x_2	010
x_3	01
x_4	10

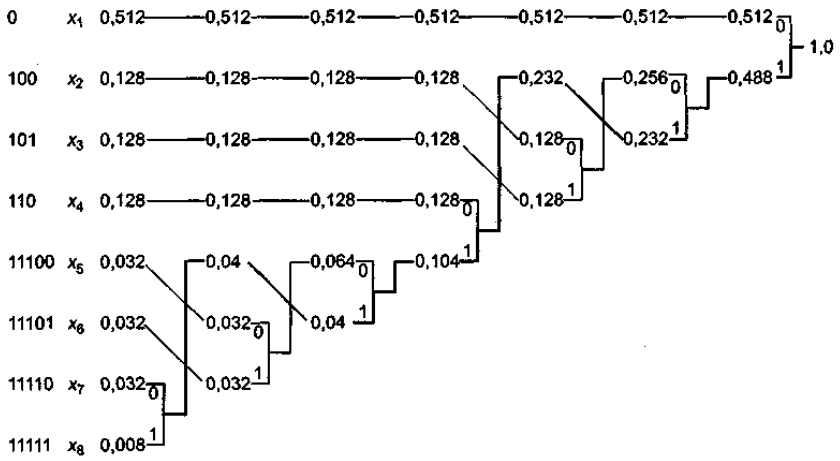
Двоичная последовательность 010 может соответствовать одному из трех сообщений:

x_2, x_3x_1, x_1x_4

Назначение третьего требования легко понять, если отметить, что при нарушении данного условия, поменяв два кода, вы сможете получить меньшее значение средней длины. Чтобы понять суть последнего

требования, предположим, что $L_N > L_{N-1}$ По второму требованию кодовое слово N - 1 не может быть префиксом кодового слова N. Поэтому первые N- 1 цифры кодового слова N составляют уникальное кодовое слово, а остальные цифры можно отбросить. Если эти два кодовых слова различаются в каком-либо бите, кроме последнего, мы можем отбросить последний бит у каждого слова, получая таким образом лучший код.

На основании этих методов можно построить код, называемый кодом Хаффман. Начнем с упорядочивания всех символов по убыванию вероятности, так что мы получим символы $(a_1, a_2,..., a_N)$ с вероятностями $P(a_1) >= P(a_2) >= ... >= P(a_N)$. Затем объединим два последних символа в эквивалентный символ с вероятностью $P(a_{N-1}) + P(a_N)$. Коды этих двух символов будут одинаковыми, кроме последних двух цифр. Теперь у нас есть новый набор из N- 1 символов. При необходимости поменяем порядок следования символов таким образом, чтобы получить символы $(b_1, b_2, ... b_{N-1})$ с вероятностями $P(b_1) >= P(b_2) >= ... >= P(b_{N-1})$. Теперь мы можем повторять этот процесс до тех пор, пока не останется всего два символа. Рисунок ниже иллюстрирует этот процесс для набора символов, заданного в начале раздела. Будем считать каждый символ листовым узлом создаваемого дерева. Объединим два узла с минимальной вероятностью в узел, вероятность которого равна сумме двух соответствующих вероятностей. На каждом шаге будем повторять этот процесс до тех пор, пока не останется только один узел. В результате мы получим дерево, в котором у каждого узла, кроме корневого, одна ветвь отходит направо и две налево. На каждом узле пометим две левые ветви символами 0 и 1 соответственно. Для каждого символа кодовое слово представляет собой строку меток от корневого узла к исходному символу. Все получившиеся в результате кодовые слова показаны в левой части рисунка. Для примера процесс построения кодового слова для символа x_7 обозначен жирной линией.



В таблице ниже приведены свойства кода Хаффмана для данного примера. Средняя длина кодового слова представляет собой вычисляемую следующим образом ожидаемую величину:

$$E[L] = \sum_{i=1}^{8} L_i P_i = 2{,}184.$$

Здесь L_i представляет собой длину i-го кодового слова. Таким образом, например, для сообщений, состоящих из 1000 символов, средняя длина кодированного сообщения равна 2184 бит. При простом кодировании каждого символа тремя битами длина этого сообщения будет составлять 3000 бит.

Символ	Код	P_i	L_i	P_iL_i	$\log(1/P_i)$	$P_i\log(1/P_i)$
x_1	0	0,512	1	0,512	0,966	0,494
x_2	100	0,128	3	0,384	2,966	0,38
x_3	101	0,128	3	0,384	2,966	0,38
x_4	110	0,128	3	0,384	2,966	0,38
x_5	11100	0,032	5	0,16	4,966	0,159
x_6	11101	0,032	5	0,16	4,966	0,159
x_7	11110	0,032	5	0,16	4,966	0,159
x_8	11111	0,008	5	0,04	6,966	0,056
		Средняя длина = 2,184			Энтропия = 2,176	

Энтропия и эффективность кодирования

Рассмотрим случайную переменную с множеством возможных значений (x_1 x_2,..., x_N), принимаемых с вероятностями (P_1, P_2,..., P_N), где P, означает вероятность результата x_i. Определим нижнюю границу средней длины кода. Мы знаем, что мерой информации для x_i является $log(1/P)$. Поэтому в идеальном случае мы сможем представить значение x_i- кодовым словом длины $L_i = log(l/P_i)$ бит. Однако в большинстве случаев $log(l/P_i)$ не является целым числом, и лучшее, что мы можем сделать, — это выбрать ближайшее целое число L_i такое что

$$\log(1/P_i) \le L_i < \log(1/P_i) + 1.$$

Умножая на P_i и суммируя по всем кодовым словам, получаем:

$$\sum_{i=1}^{N} P_i \log(1/P_i) \le \sum_{i=1}^{N} P_i L_i < \sum_{i=1}^{N} P_i \log(1/P_i) + \sum_{i=1}^{N} P_i,$$

$$H(X) \le E[L] < H(X) + 1.$$

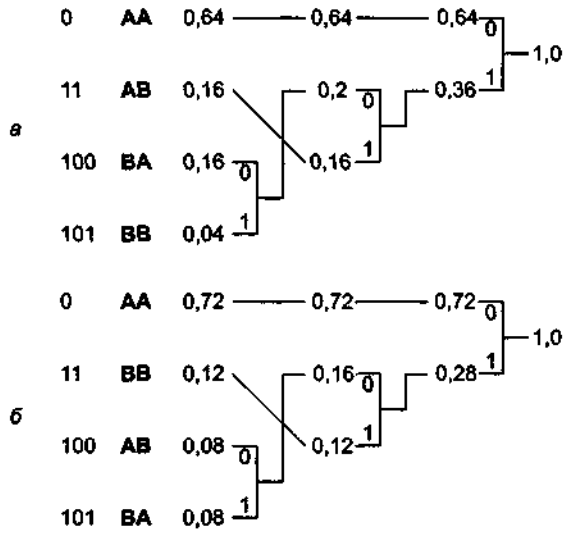
Таким образом, оптимальный код позволяет получить среднюю длину кода, не более чем на 1 бит превосходящую энтропию оригинального набора символов. Таким образом, энтропия случайной переменной X может интерпретироваться как минимальное среднее число битов, необходимых для отображения одного значения X.

Можно показать, что код Хаффмана удовлетворяет данному неравенству. Пример приводится в таблице. Средняя длина кода составляет 2,184, а энтропия равна 2,167. Обратите внимание на то, что не все отдельные кодовые слова удовлетворяют неравенству. Однако в среднем это неравенство выполняется.

Объединение символов

Предположим, что у нас есть источник, генерирующий символы из алфавита X, состоящего всего из двух символов A и B, встречающихся с вероятностью 0,8 и 0,2. Энтропия при такой схеме составляет 0,8 log(l,25) + 0,2 log(5) = 0,722. Но лучшее, что можно сделать в данном случае, — это использовать по одному биту для каждого кодового слова (например, A = 0, B = 1). Если определить эффективность кода как отношение энтропии источника (среднее число битов информации в символе) к средней длине кодового слова (среднее число битов, используемых для кодирования одного символа), тогда эффективность этого кода будет равна 0,722.

Эффективность кода можно повысить, если объединить символы в блоки и кодировать эти блоки символов. Например, если мы будем кодировать по два символа одновременно, тогда можно рассматривать полученные в результате блоки символов как новый алфавит Y, состоящий из четырех символов (AA, AB, BA, BB). Если символы алфавита X следуют в сообщениях независимо друг от друга, тогда вероятности, с которыми в них встречаются символы алфавита Y, будут равны: $P_{AA} = 0{,}64$; $P_{AB} = 0{,}16$; $P_{BA} = 0{,}16$; $P_{BB} = 0{,}04$. На рисунке показан код Хаффмана для такого варианта объединения символов в блоки, а в верхней части таблицы ниже статистика этого кода. В результате мы получаем код со средней длиной 1,56 при энтропии, равной 1,444, таким образом, эффективность кода составляет 0,926, что значительно лучше, чем при кодировании исходных символов без группировки. Обратите внимание на то, что информация источника не изменилась. Энтропия случайной переменной X равна 0,722, а энтропия К равна 0,722 • 2 = 1,444, то есть все также по 0,722 бита на символ. Еще лучших результатов можно достичь, объединив символы по три. В данном случае эффективность кода составит 2,167/2,128 = 0,992. Энтропия все также будет составлять 0,722 бит на символ (2,167/3).



Символ	Код	P_i	L_i	P_iL_i	$\log(1/P_i)$	$P_i\log(1/P_i)$
Независимые символы						
AA	0	0,64	1	0,64	0,644	0,412
AB	11	0,16	2	0,32	2,644	0,423
BA	100	0,16	3	0,48	2,644	0,423
BB	101	0,04	3	0,12	4,644	0,186
		Средняя длина = 1,56			Энтропия = 1,444	
Зависимые символы						
AA	0	0,72	1	0,72	0,474	0,341
BB	11	0,12	2	0,24	3,059	0,367
AB	100	0,08	3	0,24	3,644	0,292
BA	101	0,08	3	0,24	3,644	0,292
		Средняя длина = 1,44			Энтропия = 1,292	
Зависимость символов не учитывается						
AA	0	0,72	1	0,72		
AB	11	0,08	2	0,16		
BA	100	0,08	3	0,24		
BB	101	0,12	3	0,36		
		Средняя длина = 1,48				

Можно показать, что эффективность оптимального кода для независимых последовательностей символов может быть повышена путем объединения символов, как это было показано ранее. Для блока из K символов мы получим следующее соотношение:

$$H(X) \le \frac{E[L]}{K} < H(X) + \frac{1}{K}.$$

Таким образом, среднее количество битов на одно значение случайной переменной X можно сделать сколь угодно близким к энтропии, выбирая все большие размеры блоков.

Переходные зависимости

В предыдущем пункте предполагалось, что следующий символ в последовательности не зависит от предыдущего символа. Однако если такая зависимость существует, тогда вероятности, связанные с блоками символов, меняются. Например, рассмотрим снова алфавит X из двух символов и предположим, что к нему применимы следующие переходные вероятности:

	A	B
A	0,9	0,1
B	0,4	0,6

Таким образом, вероятность того, что за символом A последует символ A, равна 0,9; вероятность того, что за символом B последует символ A, равна 0,1 и т. д. Несложно доказать непротиворечивость

матрицы переходов при вероятностях появления отдельных символов A и B, равных 0,8 и 0,2 соответственно.

Теперь разработаем код для алфавита *Y,* составленного из комбинаций этих символов по два. Мы получим:

$$P_{AA} = \Pr(A|A)P_A = 0{,}9 \cdot 0{,}8 = 0{,}72,$$
$$P_{AB} = \Pr(B|A)P_A = 0{,}1 \cdot 0{,}8 = 0{,}08,$$
$$P_{BA} = \Pr(A|B)P_B = 0{,}4 \cdot 0{,}2 = 0{,}08,$$
$$P_{BB} = \Pr(B|B)P_B = 0{,}6 \cdot 0{,}2 = 0{,}12.$$

На уже рассмотренном рисунке (нижний граф) показан код Хаффмана для данного случая, а в средней части таблицы — полученная в результате статистика. Обратите внимание на то, что энтропия Y меньше, чем в случае, когда следующие друг за другом символы независимы (1,292 против 1,444). Поскольку вероятности для отдельных символов остались теми же самыми (P_A = 0,8, P_B = 0,2), то чем это объясняется? Ответ состоит в том, что вероятности переходов добавляют информацию. Когда становится известен первый символ последовательности, мы получаем больше информации о том, какой символ появится следующим. Поэтому неопределенность уменьшается, и суммарная энтропия также снижается. Другими словами, в последовательностях появляется избыточность. Так, например, существует значительная избыточность в предложениях на английском языке.

Для случая зависимых символов средняя длина кода равна 1,44, что дает эффективность 1,292/1,44 = 0,897. В нижней части таблицы показано, что произойдет, если проигнорировать переходные вероятности и предположить, что следующие друг за другом символы независимы даже в том случае, если они зависимы. В этом случае используются те же коды, что и для независимых символов (AA = 0, AB =11, BA= 100, BB =101). Однако теперь для второй по частоте комбинации BB используется самое длинное кодовое слово 101, поэтому средняя длина кодового слова оказывается равной 1,48, что больше средней длины, получавшейся при использовании оптимального кода Хаффмана.

Из этого обсуждения можно сделать вывод, что большей эффективности в сжатии данных можно достичь, кодируя данные большими блоками, а также учитывая зависимости данных.

Лекция 9. Сжатие без потерь

Принцип сжатия данных довольно прост. Практически все формы представления данных (текст, числа, изображение, видео) содержат избыточные элементы. Данные могут быть сжаты с одной из двух целей: для экономии места при хранении или для снижения потребностей в пропускной способности. Когда сжатые данные извлекаются из места хранения или прибывают по линии связи, должна существовать возможность распаковать данные, восстановив их исходную форму. Для этого при сжатии данных удаляемые элементы данных должны замещаться другими таким образом, чтобы получатель смог восстановить исходную информацию.

Существует две основные разновидности сжатия данных: сжатие без потерь и сжатие с потерями. При *сжатии без потерь* (lossless compression) информация не теряется и распакованные данные идентичны исходным несжатым данным. Как было показано в главе 19, эффективность сжатия без потерь ограничена энтропией источника данных. В случае *сжатия с потерями* (lossy compression) распакованные данные могут быть приемлемым приближением (в соответствии с некоторым критерием точности) к исходным несжатым данным. Например, при сжатии изображений или видеоданных критерий может заключаться в том, что для человеческого глаза распакованное изображение неотличимо от оригинала.

Важным фактором при разработке и выборе алгоритмов сжатия данных является объем работ, требуемый для сжатия и распаковки данных. В общем случае существует компромисс между достижимой степенью сжатия и скоростью работы алгоритма сжатия, а также стоимостью этой работы.

Методы группового кодирования

Групповое кодирование (run-length encoding) представляет собой простой метод сжатия данных без потерь, довольно эффективный для сжатия текста. Этот метод также находит применение в факсимильном сжатии.

Подавление нулей

Один из старейших и простейших методов сжатия данных известен как подавление нулей (null suppression), или подавление пробелов (blank suppression). Он до сих пор применяется в протоколе уровня передачи данных BISYNC (Binary Synchronous Communications protocol — двоичный синхронный протокол связи) IBM 3780. В тексте или потоке символов часто встречаются длинные строки пробелов или нулей. В методе подавления нулей передатчик сканирует данные в поиске строк пробелов и заменяет каждую такую строку двухсимвольным кодом. Код состоит из специального управляющего символа, за которым следует число пробелов в строке. Например, пусть имеется код с символами пробела, обозначенными знаком b:

XYZbbbbbQRX.

Эта строка заменяется следующей строкой, в которой S_c представляет собой специальный управляющий символ:

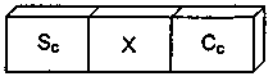
XYZS_c5QRX.

Такая схема позволяет сделать короче все строки из трех и более пробелов. В методе подавления нулей приемник ищет в потоке входящих символов специальный символ, используемый для индикации удаленных пробелов. Получив такой символ, получатель понимает, что следующий символ содержит количество удаленных пробелов. По этой информации может быть восстановлен исходный поток данных. При том что метод подавления нулей представляет собой крайне примитивную форму сжатия данных, его преимущество заключается в том, что он очень легко реализуется. Кроме того, выигрыш даже от применения такого простого метода может быть значительным. Как сообщалось в [113], выигрыш от перехода с протокола IBM 2780 на протокол 3780 в ряде компьютерных конфигураций составил от 30 до 50%, и все это благодаря данному методу сжатия данных.

Групповое кодирование

Групповое кодирование представляет собой обобщение метода подавления нулей. Групповое кодирование применяется для сжатия повторяющихся данных любого типа. На рисунке иллюстрируется применение данного метода к символьным данным. Здесь используются следующие обозначения:

- S_c — специальный символ, указывающий на то, что за ним следуют сжатые данные;
- X — любой повторяющийся символ;
- C_c — счетчик символов, то есть количество повторений сжатого символа.



Примеры сжатия при групповом кодировании

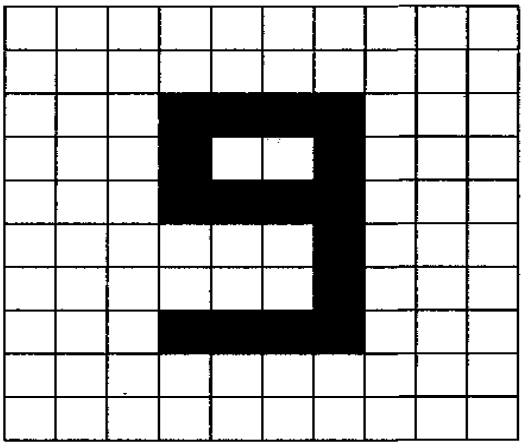
Исходная строка данных	Кодированная строка данных
$******55.72	$$S_c$*655.72
---------	S_c-9
GunsbbbbbbbbbButter	GunsS_cb9Butter

Как и в методе подавления нулей, передатчик ищет последовательности повторяющихся символов. В данном случае он заменяет их трехсимвольным кодом. Код состоит из специального индикатора сжатия, за которым следуют сам повторяющийся символ и число его повторений. Таким образом, данный метод позволяет сократить место, занимаемое любой последовательностью из четырех и более одинаковых символов.

Эффективность метода группового кодирования зависит от того, насколько часто в исходных данных встречаются последовательности повторяющихся символов, и от средней длины таких серий.

Стандартной мерой эффективности сжатия является коэффициент сжатия, представляющий собой отношение длины несжатых данных к длине сжатых данных (включая символы кодирования).

Групповое кодирование было одним из первых методов сжатия факсимильных сообщений, но теперь оно более не используется для этой цели. Однако этот метод следует изучить, так как он применяется в более сложных методах сжатия изображений. При использовании метода группового кодирования для изображений вместо отсканированной и оцифрованной линии передаются длины серий белых и черных элементов изображения. На рис. 20.2 показан пример применения метода группового кодирования к простому изображению размером 10x10 точек. Это может быть факсимильное изображение или отсканированное растровое компьютерное изображение. Изображение формата 10 x 10 легко преобразуется в 100-битовый код. В данном примере каждый пиксел представляется одним битом, обозначающим белый или черный цвет. Код длин серий состоит из длин чередующихся черных и белых последовательностей. Поскольку при таком кодировании изображения черный и белый цвета всегда чередуются, нет необходимости в использовании специального символа, указывающего цвет серии. Таким образом, кодированный поток данных представляет собой строку чисел, обозначающих длины чередующихся серий черных и белых точек. Обратите внимание на то, что в данном простом примере кодированные данные занимают больше места, чем исходные. Однако в случае применения данного метода к типичной странице текста этот метод будет сжимать данные. Тем не менее, это далеко не самый эффективный способ сжатия изображений.



Изображение

0000000000
0000000000
0001111000
0001001000
0001111000
0000001000
0000001000
0001111000
0000000000
0000000000

Длина равна 100 бит

Двоичный код

23W 4B 6W 1B 2W 1B 6W 4B 9W
1B 9W 1B 6W 4B 23W

или

23 4 6 1 2 1 6 4 9 1 6 4 23

Длина равна 15 символов, или 120 бит

Простое групповое кодирование

Факсимильное сжатие

Методы сжатия данных очень важны для широкого применения цифровых факсимильных аппаратов. Для примера рассмотрим типичную страницу, отсканированную с разрешением в 200 пелов (белых или черных точек) на дюйм (что является приемлемой, но не высокой степенью разрешения). В результате такая страница содержит 3 740 000 бит (8,5 дюймов x 11 дюймов x 40 000 пелов на квадратный дюйм). При базовой скорости службы ISDN в 64 Кбит/с передача такой страницы займет около одной минуты. Обычно пользователи ожидают от систем работы со скоростью, близкой к скорости копирования, то есть несколько секунд на страницу. Для удовлетворения данного требования, если не прибегать к увеличению скорости передачи данных в линии, необходимо использовать метод сжатия данных.

Сектор ITU-T стандартизировал два метода сжатия данных без потерь для факсимильной связи: модифицированный код Хаффмана и модифицированный код READ (Relative Element Address Designate — относительное назначение адресов элементов). Модифицированный код Хаффмана используется по умолчанию в факсимильной аппаратуре группы 3, в которой модифицированный код READ может применяться факультативно. В аппаратуре группы 4 применяется модифицированный код READ. Кратко эти два стандарта (группы 3 и 4) характеризуются следующим образом:

- *Группа 3*. Первый цифровой факсимильный стандарт. Эта система обеспечивает только кодирование черных и белых значений с плотностью сканирования в 200 точек на дюйм по горизонтали и от 100 до 200 по вертикали. В аппаратуре группы 3 используются цифровая схема кодирования, а также средства уменьшения избыточной информации в сигнале документа перед модуляцией. Предполагается, что передача сигнала осуществляется через модем по аналоговой телефонной линии. Передача данных ускоряется в три и более раз по сравнению с группой 2.
- *Группа 4*. Также представляет собой черно-белый цифровой факсимильный стандарт. Эта группа предназначена для использования в цифровых сетях со скоростями до 64 Кбит/с при условии безошибочного приема. Стандартизированы разрешения от 200 до 400 точек на дюйм. Как и в группе 3, для снижения количества передаваемых битов применяются методы сжатия. В аппаратуре группы 4 время передачи одной страницы сокращено до нескольких секунд по сравнению с несколькими минутами в предыдущих стандартах.

Модифицированный код Хаффмана

В типичном документе черные и белые области имеют тенденцию к объединению. Если рассматривать документ как последовательность линий и обратить внимание на расположение участков белого и черного в линии, можно обнаружить длинные серии белых и черных точек. Благодаря этому свойству можно предположить, что сжатие на основе метода группового кодирования даст хороший результат. Состоящие из двух значений входные данные преобразуются в длины серий, которые затем кодируются для передачи. Кроме того, поскольку в общем случае длинные серии черных или белых точек менее вероятны, чем короткие, можно воспользоваться преимуществом кодирования последовательностей переменной длины. Для кодирования факсимильных документов может применяться код Хаффмана, описанный ранее. Этот метод можно применить к изображению построчно, кодируя последовательности черных и белых точек. Например, предположим, что при сканировании отдельной строки получается следующая последовательность черных и белых точек: W7, B7, W4, B8, W4, B7, W10 (здесь W означает белый, а B — черный). Если рассматривать каждый из этих элементов как символ алфавита источника, тогда для кодирования этих данных может быть использован метод кодирования Хаффмана. Однако поскольку стандарт ITU-T требует по меньшей мере 1728 точек на линию, количество различных кодов, а следовательно, и средняя длина кода будет очень большой. Альтернативой является модифицированный метод кодирования Хаффмана. В этом методе длина серии *N* рассматривается как сумма двух слагаемых:

$$N = 64m + n;\ m = 0, 1, 2, ..., 27;\ n = 0, 1, 2, ..., 63.$$

То есть длина каждой серии черных или белых точек считается величиной, кратной 64 с остатком.

Теперь каждая длина серии может быть представлена двумя значениями, одним для *m* и другим для *n*, и эти значения могут кодироваться при помощи метода Хаффмана. Например, строка из 200 черных точек подряд может быть выражена как 64*3 + 8. Для этого сектором ITU-T были определены восемь образцов документов и рассчитаны вероятности нахождения в документах серий различных длин. Поскольку для серий из черных и белых точек эти вероятности различаются, были сосчитаны два множества вероятностей. Длина серии делится на 64, после чего частное и остаток от деления кодируются двумя кодовыми словами. Если длина серии меньше 64 (частное от деления равно нулю), то такая длина серии кодируется только кодовым словом остатка. Серии длиной более 64 точек кодируются двумя кодами: кодом остатка (*n*) и кодом кратности *(m)*. Имеется еще несколько деталей, касающихся этого кода. Каждая строка заканчивается уникальным кодовым словом EOL (End Of Line — конец строки). Это кодовое слово, никогда не встречающееся в строках данных, обеспечивает восстановление синхронизации в случае ошибок. Внутри строки должны чередоваться кодовые слова для белых и черных серий. Обратите внимание на то, что для белых и черных серий используются различные кодовые слова. Это обеспечивает дополнительную защиту от ошибок. Наконец, по соглашению каждая строка начинается с белой серии. Если первая точка в линии черная, то используется белая серия нулевой длины.

Арифметическое кодирование

Метод кодирования Хаффмана хотя и является простым в реализации и относительно эффективным, тем не менее, не позволяет достичь максимального коэффициента сжатия, если только все вероятности не представляют собой отрицательных степеней числа 2. Арифметическое кодирование предназначено для достижения более высоких коэффициентов сжатия за счет усложнения вычислений. При этом изображение обрабатывается таким образом, чтобы вероятности приближались к отрицательным степеням числа 2. Арифметическое кодирование используется в стандартах JPEG и MPEG. Мы начнем с концепции, лежащей в основе арифметического кодирования, а затем рассмотрим некоторые детали.

Основная концепция

Вспомним из обсуждения метода кодирования Хаффмана, что если отдельные кодируемые символы встречаются в исходном тексте с вероятностью P_i, то в лучшем случае мы можем использовать для каждого символа код длины L_i, удовлетворяющей следующему неравенству:

$$\log(1/P_i) \le L_i < \log(1/P_i) + 1.$$

Здесь P_i представляет собой вероятность, с которой i-й символ встречается в исходных данных, а L_i — длину двоичного кода, которым кодируется данный символ. Исходя из данного неравенства, можно показать, что

$$H(X) \le E[L] < H(X) + 1.$$

Здесь $H(X)$ представляет собой энтропию множества символов X, а $E[L]$ является средней длиной кода для множества X. В схеме сжатия без потерь $H(X)$ всегда будет нижней границей объема сжатых данных, так как $H(X)$ определяет среднее количество информации, содержащейся в символьном наборе. Наконец, было показано, что эффективность алгоритма сжатия может быть повышена путем объединения символов и одновременного кодирования блоков по K символов каждый. В результате мы получаем следующее неравенство:

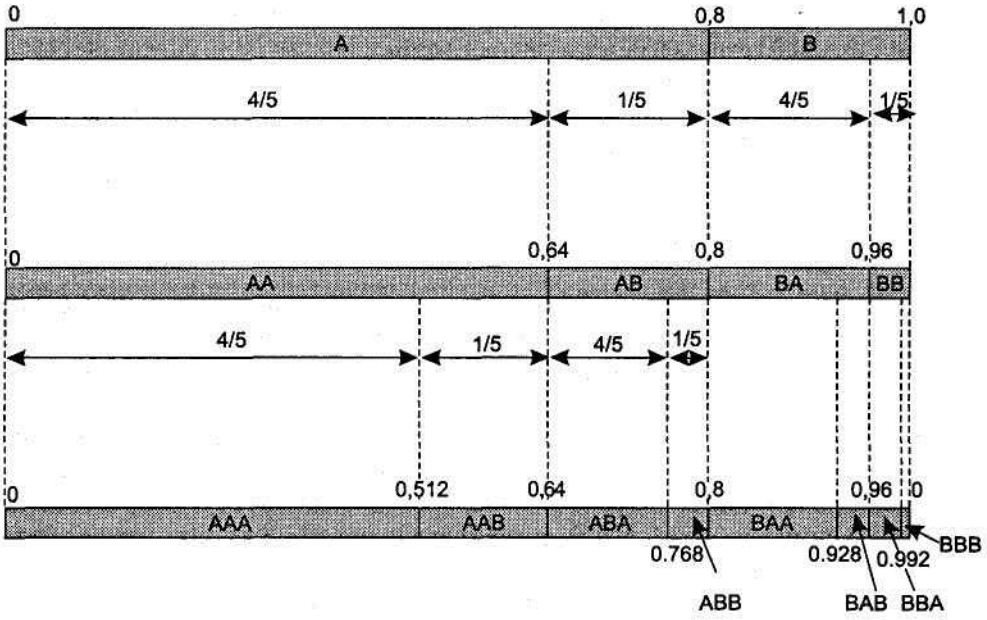
$$H(X) \le \frac{E[L]}{K} < H(X) + \frac{1}{K}.$$

Таким образом, существует способ повышения эффективности алгоритма сжатия. Если нам нужно отправить сообщение длиной в N символов, мы можем использовать блок длиной N. Таким образом, мы обращаемся с каждым сообщением как с одним из M^N возможных результатов, где M представляет собой количество атомарных символов, а N — длину сообщения. В качестве примера рассмотрим случай, обсуждавшийся ранее, в котором имеются два символа, A и B, встречающиеся в исходном тексте с вероятностями 0,8 и 0,2. Если мы передаем сообщение длиной в 3 символа ($M = 2$, $N = 3$), тогда существуют $2^3 = 8$ возможных результатов. Если появление в потоке данных каждого символа не зависит от предыдущих символов, тогда мы можем написать вероятности всех 8 результатов:

$P_{AAA} = 0{,}512$, $P_{BAA} = 0{,}128$,
$P_{AAB} = 0{,}128$, $P_{BAB} = 0{,}032$,
$P_{ABA} = 0{,}128$, $P_{BBA} = 0{,}032$,
$P_{ABB} = 0{,}032$, $P_{BBB} = 0{,}008$.

Один из возможных методов состоит в применении кода. Для сообщения с восемью возможными результатами это разумно, но при очень длинных сообщениях использование этого метода приведет к большим вычислительным затратам. Метод арифметического кодирования предоставляет остроумную альтернативу. Заметим, что сумма вероятностей всех возможных сообщений должна быть равна 1,0. Таким образом, можно разложить все результаты на интервале [0,1] в виде отрезков, длины которых равны их вероятностям. На рисунке также показан простой метод генерирования интервалов по одному символу. Эта процедура выглядит следующим образом:

1. Алгоритм начинается с позиции первого символа в сообщении, а интервал делится в соответствии с вероятностями отдельных символов. В данном случае имеется два символа, A и B, встречающиеся в исходном тексте с вероятностями 0,8 и 0,2. Поэтому интервал [0, 1] разбивается на подынтервалы [0, 0,8) и [0,8, 1). Граница между интервалами может быть отнесена произвольно к любому из интервалов.
2. Каждый подынтервал разбивается тем же способом, который применялся на шаге 1. То есть каждый подынтервал дробится на доли в соотношении 4:1.
3. Этот процесс повторяется для N символьных позиций (сообщение длины N).



Теперь результату каждого сообщения поставлен в соответствие интервал, пропорциональный вероятности сообщения. Таким образом, каждый результат может быть представлен двоичным дробным числом, равным некоторой точке на интервале. Будем использовать для этого нижнюю границу каждого интервала. Результаты такой операции показаны в пятой колонке таблицы ниже, при этом используются пять значащих цифр правее двоичной запятой. Такой точности достаточно для однозначной идентификации каждого интервала, а следовательно, и каждого результата. Поэтому мы можем использовать дробную часть каждого значения в качестве кода для соответствующего результата (например, ABA= 10100). Однако полученные коды можно усовершенствовать. Например, можно удалить из кода некоторые цифры, если только это не приведет к неразличимости кодов (то есть ни один код не должен совпадать с начальными цифрами другого кода). Результат этой операции показан в шестой колонке таблицы. Обратите внимание на то, что в общем случае более короткие интервалы идентифицируются большим числом битов. В этом есть смысл: нижняя граница короткого интервала близка к нижней границе следующего интервала. Поэтому двоичные дроби для этих двух границ будут совпадать в нескольких разрядах. Чем меньше интервал, тем больше одинаковых разрядов в соответствующих дробях.

Последовательность	**P_i**	**Кумулятивная Вероятность**	**Интервал**	**Двоичное представление нижней границы**	**Код**	**$P_i L_i$**	**g(**
AAA	0.512	0.512	[0,0.512]	0.00000	0	0.512	4
AAB	0.128	0.64	[0.512,0.64]	0.10000	100	0.384	
ABA	0.128	0.768	[0.64, 0.768]	0.10100	101	0.384	
ABB	0.032	0.8	[0.768,0.8]	0.11000	1000	0.16	9
BAA	0.128	0.928	[0.8,0.928]	0.11001	1001	0.64	
BAB	0.032	0.96	[0.928,0.96]	0.11101	111	0.096	9
BBA	0.032	0.992	[0.96, 0.992]	0.11110	1110	0.096	9
BBB	0.008	1.0	[0.992,1.0]	0.11111	1111	0.04	6
						2.408	**7**

Таким образом, мы разработали новый метод назначения кодов сообщениям. В последних двух колонках таблицы средняя длина нового кода сравнивается с энтропией множества сообщений. Несмотря на всю свою внешнюю привлекательность новый метод обладает двумя существенными недостатками:

- Если все возможные интервалы вычислять заранее, то объем необходимых вычислений будет огромен. Например, если длина сообщения равна 1000 символов и используется алфавит из 128 символов (например, набор символов ASCII), тогда число всех возможных сообщений будет составлять $128^{1000} = 10^{2107}$.
- Описанная схема применима только к сообщениям фиксированной длины. Метод арифметического кодирования позволяет устранить оба недостатка.

Чистое арифметическое кодирование

Мы начнем с простейшей формы алгоритма арифметического кодирования. Хотя этот алгоритм преодолевает упомянутые ранее недостатки, в нем сохраняются некоторые вычислительные сложности. Тем не менее, проще понять метод арифметического кодирования, рассмотрев сначала более простую форму. При арифметическом кодировании нет необходимости заранее генерировать интервалы для всех возможных сообщений. Вместо этого интервалы для отдельного сообщения вычисляются посимвольно. Таким образом, для передачи одного сообщения вычисляется только один интервал. Кроме того, поскольку процесс является посимвольным, нет необходимости фиксировать длину сообщения заранее. Вычисления продолжаются до тех пор, пока не будет достигнут конец сообщения.

Базовый алгоритм

Каждый шаг алгоритма начинается с полуоткрытого интервала *[L, H),* вначале инициализируемого как [0,1):

1. Текущий интервал разбивается на подынтервалы по одному для каждого возможного символа. Каждый подынтервал пропорционален вероятности, с которой соответствующий символ встречается в тексте.
2. Выбирается подынтервал, соответствующий текущему символу. Этот по дынтервал становится текущим интервалом.
3. Если обработано все сообщение, выполняется следующий шаг, если нет, воз вращаемся к шагу 1.
4. Выводится количество битов, достаточное для того, чтобы можно было от личить данный интервал от двух соседних интервалов.
5. Выводится некий специальный код конца сообщения, чтобы уведомить по лучателя.

На первом шаге нет необходимости вычислять все подынтервалы. Вместо этого вычисляются только подынтервал, соответствующий текущему символу. Кратко эта операция описана далее. Введем следующие обозначения:

- X — набор символов $(x_1, x_2, ..., x_M)$;
- $X(k)$ — символ, встретившийся на k-й итерации, то есть k-й символ сообщения;
- $I(k)$ — значение индекса $X(k)$, например, если $X(k) = x_3$, тогда $I(k) = 3$;
- M — количество возможных символов;
- $P_X(i) = \Pr[X = i] = P_i$ — функция плотности вероятности для X;
- $F_X(i) = \sum_{j=1}^{i} P_j$ — кумулятивная функция распределения вероятностей для X;
- L_k — нижняя граница подынтервала после k-й итерации алгоритма;
- H_k — верхняя граница подынтервала после k-й итерации алгоритма;
- $W_k = H_k - L_k$ — ширина подынтервала после k-й итерации алгоритма.

При этом каждая итерация алгоритма включает следующие вычисления:

$$L_k = L_{k-1} + F_X(I(k) - 1)W_{k-1},$$
$$H_k = L_{k-1} + F_X(I(k))W_{k-1},$$
$$W_k = H_k - L_k = \Pr[X = X(k)]W_{k-1}.$$

Ширина конечного подынтервала равна произведению вероятностей каждого встретившегося за время работы алгоритма символа и поэтому представляет собой вероятность того, что данное сообщение *m* встретится в M^N возможных сообщениях. Это можно выразить так:

$$\Pr(m) = \prod_{k=1}^{N} \Pr[X = X(k)].$$

Можно показать, что для уникальной идентификации интервала сообщения *m* требуется, самое большее, *2+log(1/Pr(m))* бит. Чтобы продемонстрировать это на интуитивно понятном уровне, рассмотрим два первых интервала с соответствующими длинами Pr(1) и Pr(2), то есть интервалы [0, Pr(1))и [Pr(1), Pr(1) + Pr(2)). Теперь предположим, что для определения интервала мы используем его нижнюю границу. При этом двоичная дробь, представляющая Pr(1), будет кодом для второго интервала. Существует некоторое число *j,* такое, что 2^{-j} будет наименьшей величиной, удовлетворяющей неравенству 2^{-j} > Pr(1). Это число представляет собой двоичную дробь, начинающуюся с (log(l/Pr(m)) - 1) бит, за которыми следует 1. При использовании этого значения в качестве кода для второго интервала гарантируется, что этот код уникально идентифицируется по отношению к первому интервалу. Обратите внимание на то, что в данной формулировке не делается предположений о стационарности или независимости переходов.

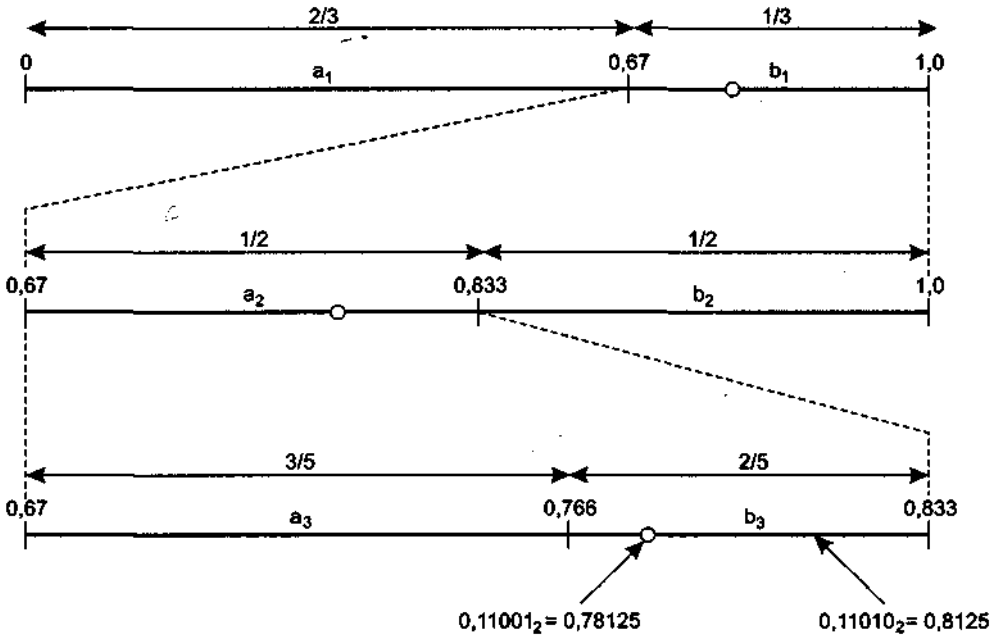
Пример

Пусть имеются два символа, a и b, вероятности которых зависят от положения в трехсимвольном общении:

$\Pr(a_1) = 2/3$, $\Pr(b_1) = 1/3$,
$\Pr(a_2) = 1/2$, $\Pr(b_2) = 1/2$,
$\Pr(a_3) = 3/5$, $\Pr(b_3) = 2/5$.

Здесь $\Pr(a_i)$ представляет собой вероятность того, что символ a встретится в i-й позиции сообщения. В данном случае кодируется сообщение «bab».

На рисунке ниже показано арифметическое кодирование этого сообщения. Данному сообщению соответствует конечный интервал [23/30,5/6) = [0,766,0,833). В двоичном виде конечный интервал выглядит следующим образом: [0,110001..., 0,110101...). Обратите внимание на то, что все числа, начинающиеся с 0,110001, целиком попадают в интервал, но существуют некоторые числа, начинающиеся с 0,1100, не попадающие в этот интервал (например, 0,1100001 < 23/30). Поэтому кода 11001 достаточно, чтобы однозначно идентифицировать интервал и, следовательно, однозначно идентифицировать сообщение. В общем случае, чем меньше конечный интервал (соответствующий менее вероятному сообщению), тем больше битов кода требуется для уникальной идентификации интервала.



Декодирование

Процесс декодирования выполняется по той же общей поэтапной схеме. На этот раз обнаруживаются последовательные подынтервалы, приводящие к конечному интервалу и открытию соответствующих сообщений. Общие правила выглядят следующим образом. Опять же, начнем с интервала [0,1):

1. Разделим текущий интервал на подынтервалы, по одному для каждого сим вола. Длина каждого подынтервала пропорциональна вероятности появле ния соответствующего символа.

2. Выберем подынтервал, содержащий код, выраженный в виде двоичной дро би. Выведем символ, определяющий этот код.

Эти два шага повторяются до тех пор, пока не будет распаковано все сообщение. Существует несколько способов определить, что сообщение распаковано полностью. К исходному сообщению может быть добавлен символ конца сообщения. В этом случае процесс декодирования остановится при обнаружении этого символа. Другой способ состоит в том, чтобы при декодировании на каждом шаге проверять, все ли двоичные дроби, начинающиеся с кода, попадают в один интервал, и прекратить декодирование, когда это условие перестанет выполняться.

Процесс декодирования, как и процесс кодирования, иллюстрирует тем же рисунком. Численное значение конечного кода (0,78125 = 0,11001г) обозначается кружком на каждой линии процесса.

Инкрементное арифметическое кодирование

У только что описанного алгоритма чистого арифметического кодирования есть два недостатка. Во-первых, уменьшающиеся размеры текущего интервала требуют все более точных вычислений. Во-вторых, код не может быть послан, пока не обработано все сообщение.

Метод инкрементного арифметического кодирования позволяет решить обе проблемы. При использовании этой техники перед выполнением каждого следующего шага алгоритма текущий интервал всегда увеличивается, а кодовые биты генерируются на каждом шаге и могут передаваться или сохраняться после каждого шага.

Описание алгоритма

Каждый шаг алгоритма начинается с полуоткрытого интервала *[L, H),* который инициализируется значением [0,1). Кроме того, используется *счетчик следования* (follow counter), инициализируемый значением 0:

1. Текущий интервал разбивается на подынтервалы по одному для каждого возможного символа. Длина каждого подынтервала пропорциональна вероятности появления соответствующего символа.

2. Выбирается подынтервал, соответствующий текущему символу.

3. Следующие шаги выполняются в цикле до тех пор, пока не выполнится условие выхода из цикла:

   a. Если новый подынтервал не попадает целиком в один из следующих интервалов: [0, 0,5), [0,25, 0,75) или [0,5,1), — выйти из цикла.

   b. Если новый подынтервал попадает в интервал [0,0,5), вывести бит 0, после которого ноль или более битов 1, соответствующих значению счетчика следования, после чего значение счетчика следования установить на ноль. Удвоить размер подынтервала, линейно расширив [0, 0,5) до [0,1). То есть установить $L = 2L$ и $H = 2H$.

   c. Если новый подынтервал попадает в интервал [0,5, 1,0), вывести бит 1, после которого ноль или более битов 0, соответствующих значению счетчика следования, после чего значение счетчика следования установить на ноль. Удвоить размер подынтервала, линейно расширив [0,5,1) до [0,1). То есть установить $L = 2(1 - 0,5)$ и $H= 2(H- 0,5)$.

   d. Если новый подынтервал попадает в интервал [0,25 0,75), увеличить на единицу значение счетчика следования. Удвоить размер подынтервала,, линейно расширив [0,25, 0,75) до [0,1). То есть установить $L = 2(L - 0,25)$ и $H=2(H-0,25)$.

Шаги с 1 по 3 повторяются до тех пор, пока не будет обработано все сообщение. Вот что, по сути, выполняет данный алгоритм. Если новый подынтервал целиком попадает в интервал [0,0,5) или [0,5,1,0),

алгоритм формирует ведущий бит, указывающий, в какую половину интервала попадает новый подынтервал, после чего интервал удваивается, так что новый интервал отражает только неизвестную часть конечного интервала. Посмотрим, как это работает. Предположим, что на первом шаге формируется подынтервал в верхней половине единичного интервала Таким образом мы узнаем, что конечный подынтервал также должен находиться в верхней половине единичного интервала, и поэтому старший бит конечного кода должен быть равен 1. Когда этот бит становится известным, вторую половину единичною интервала можно проигнорировать, а подынтервал удвоить для определения следующего бита кода.

Если конечные точки текущего интервала близки к точке 0,5, но находятся по разные стороны от этой точки, тогда расположение следующего результата еще не определено. Однако каким бы он ни был, следующий бит будет иметь противоположное значение. Читатель может сам поэкспериментировать с несколькими значениями, чтобы убедиться, что это так. Поэтому алгоритм следит за следующим битом и симметрично расширяет интервал вокруг значения 0,5.

Алгоритмы совпадения строк

Такие алгоритмы, как метод Хаффмана и арифметическое кодирование, основаны на знании оценки статистики сжимаемых данных. Эти алгоритмы не адаптируются к изменениям в представлении данных. Кроме того, ограничения памяти ставят предел их способности фиксировать взаимосвязи высокого порядка между словами и фразами текста. Другой подход, доказавший свою эффективность, используется в семействе методов, известных под названием алгоритмов совпадения строк.

В то время как в методе Хаффмана и при арифметическом кодировании входные последовательности фиксированной длины преобразуются в коды переменной длины, алгоритмы совпадения строк преобразуют входные последовательности переменной длины в коды фиксированной длины. Кроме того, алгоритмы совпадения строк не выдвигают априорных предположений о статистике данных, а адаптируются к изменениям в характере входных данных по мере их обработки.

Все алгоритмы данной категории обязаны своим происхождением израильским исследователям Якову Зива (Jacob Zib) и Аврааму Лемпелю (Abraham Lempel). В 1977 г. они описали метод, основанный на буфере скользящего окна, в котором содержится только что обработанный текст. Этот алгоритм обычно называют LZ77. Разновидность этого алгоритма используется в популярной в Интернете схеме сжатия zip (PKZIP, gzip, zipit и т. д.). В следующем году те же авторы описали улучшенную версию этого алгоритма, основанного на структурированном в виде дерева словаре. Этот алгоритм называется LZ78. Затем алгоритм LZ78 был усовершенствован и назван LZW. Многие программы сжатия основаны на алгоритмах LZ78 и LZW, включая стандарт *V.42bis* сектора ITU-T, предназначенный для голосовых модемов, популярный формат сжатия изображений GIF, а также программу сжатия данных в операционной системе UNIX. В алгоритмах LZ77, LZ78 и их вариантах используется тот факт, что слова и фразы в потоке текста (элементы изображения в случае GIF) повторяются с большой вероятностью. Когда встречается повторяющаяся последовательность, она может быть заменена коротким кодом. Программа сжатия ищет подобные повторяющиеся участки текста и «на лету» формирует коды, которыми заменяет их на выходе. Те же коды могут использоваться для обозначения новых последовательностей. Алгоритм должен быть определен таким образом, чтобы программа распаковки могла найти текущее соответствие между кодами и последовательностями исходных данных.

Другой способ повышения эффективности алгоритмов совпадения строк состоит в том, чтобы учитывать энтропию входных данных, рассматриваемых в виде последовательности строк. Вспомним, что чем выше уровень агрегации входных символов, тем ниже энтропия. Поэтому следует ожидать, что, используя совпадения строк, можно добиться высоких коэффициентов сжатия.

Алгоритм LZ77

Прежде чем перейти к изучению деталей алгоритма LZ77, рассмотрим простой пример. Пусть имеется следующая бессмысленная фраза:

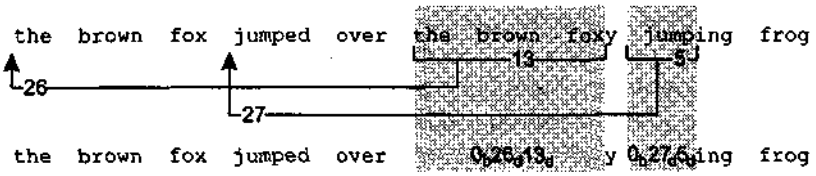
```
the brown fox jumped over the brown foxy jumping frog
```

Алгоритм обрабатывает этот текст слева направо. Вначале каждый символ преобразуется в 9-битовый код, состоящий из двоичной единицы, за которой следует 8-битовое ASCII-представление символа. Во время обработки текста алгоритм ищет повторяющиеся последовательности. Когда встречается повторяющаяся последовательность, алгоритм ищет конец такой последовательности, то есть каждый раз алгоритм отыскивает как можно больше символов. Первой такой последовательностью является the brown fox. Повторяющаяся последовательность заменяется указателем на первый экземпляр последовательности и длиной этой последовательности. В данном случае последовательность the brown fox встречается на 26 позиций раньше и имеет длину 13 символов. Для данного примера рассмотрим два варианта кодирования: 8-битовый указатель и 4-битовая длина или 12-битовый указатель и 6-битовая длина. При этом 2-битовый

заголовок указывает, который вариант выбран: 00 означает первый вариант, а 01 — второй вариант. Таким образом, второй экземпляр последовательности the brown fox кодируется как $<00_b><26_d><13_d>$ или 00 00011010 1101.

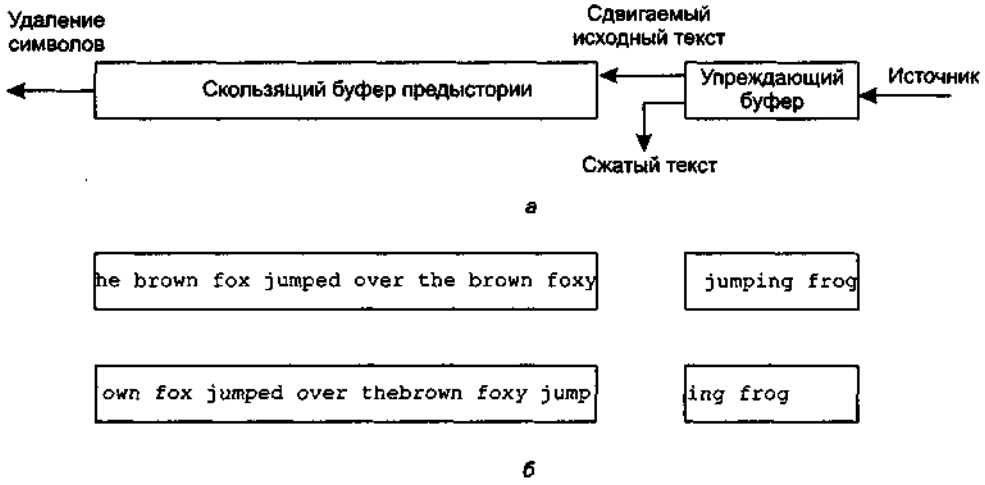
Оставшиеся части сжатого сообщения представляют собой букву y, последовательность $<00_b><27_d><5_d>$, заменяющую последовательность, состоящую из пробела, за которым следует строка jump, и последовательность символов «ing frog».

На рис. 20.7 иллюстрируется преобразование алгоритма сжатия. Сжатое сообщение состоит из 35 9-битовых символов и двух кодов, то есть всего из 35 • 9 + + 2 • 14 = 343 бит. Таким образом, при 424 бит в несжатом сообщении коэффициент сжатия данного метода в этом примере составил 1,24.



Алгоритм сжатия

В алгоритме сжатия LZ77 и его вариантах используются два буфера. В скользящем буфере предыстории хранятся *N* только что обработанных символов источника, а в упреждающем буфере содержатся следующие *L* символов, ожидающих обработки. Алгоритм пытается найти соответствие между двумя и более символами из начала упреждающего буфера и строкой в скользящем буфере предыстории. Если соответствие не обнаружено, первый символ упреждающего буфера выводится как 9-битовый символ, а также сдвигается в скользящее окно, при этом самый старый символ выбрасывается из скользящего окна. Если совпадение обнаруживается, алгоритм продолжает искать самое длинное совпадение. Затем строка, для которой найдено соответствие, выводится в виде триплета (индикатор, указатель, длина). Затем из скользящего окна выдвигаются столько символов, сколько содержится в кодированной триплетом последовательности, и столько же новых символов исходного текста помещаются в скользящее окно.



На нижней части рисунка показана работа данной схемы с последовательностью из нашего примера. В иллюстрации предполагается использование 39-символьного скользящего окна и 13-символьного упреждающего буфера. В верхней части примера первые 40 символов были обработаны, и несжатая версия последних 39 символов находится в скользящем окне. Остальная часть сообщения находится в упреждающем буфере. Алгоритм сжатия находит следующее совпадение, сдвигает 5 символов из упреждающего буфера в скользящее окно и формирует код для этой строки. Состояние буфера после этих операций показано в нижней части примера.

Несмотря на то что алгоритм LZ77 эффективен и адаптируется к природе текущего входного потока, он обладает рядом недостатков. Для поиска совпадений в предшествующем тексте этот алгоритм использует окно конечного размера. Если размер текста намного превышает размер окна, то многие потенциальные совпадения остаются необнаруженными. Размер окна может быть увеличен, но это приведет к двум нежелательным эффектам. Во-первых, время работы алгоритма возрастет. Во-вторых, придется увеличить размер поля указателя.

Алгоритм распаковки

Процесс распаковки текста, сжатого при помощи алгоритма LZ77, прост. Алгоритм распаковки должен сохранять последние N символов распакованного результата. Когда встречается закодированная строка, алгоритм распаковки заменяет код с помощью долей указателя и длины.

Алгоритмы LZ78 и LZW

Алгоритмы LZ78 и LZW пытаются обойти ограничения схемы LZ77, вместо ограниченного окна создавая словари. Как и схема LZ77, алгоритм LZW (и LZ78) поддерживает словарь строк вместе с их кодами как для сжатия, так и для распаковки. Когда любая строка словаря появляется на входе алгоритма сжатия, эта строка заменяется кодом. Распаковщик, наоборот, заменяет коды соответствующими им строками. По мере работы алгоритма сжатия к словарю добавляются новые строки.

В любой момент времени словарь содержит все односимвольные строки плюс некоторые многосимвольные строки. Механизм работает так, что любые ведущие подстроки присутствующей в словаре строки также могут использоваться в качестве элементов словаря. Например, если в словаре есть строка MEOW, снабженная своим уникальным кодовым словом, тогда строки MEO и ME также присутствуют в словаре и у каждой есть свое уникальное кодовое слово. Логически словарь может быть представлен в виде набора деревьев, при этом корень каждого дерева соответствует символу алфавита. Таким образом, по умолчанию имеется 256 деревьев (все возможные 8-битовые символы). Далее будет более детально описан алгоритм, имеющий много общего со стандартом *V.42bis* и схемами LZ78 и LZW.

Сжатие

Прежде чем перейти к описанию алгоритма, введем следующие обозначения:

- C_1 — следующее доступное неиспользуемое кодовое слово;
- C_2 — размер кодового слова, по умолчанию 9 бит;
- N_2 — максимальный размер словаря, равный количеству кодовых слов ($2^{C}{}_{2}$);
- N_3 — размер символа, по умолчанию 8 бит;
- N_5 — первое кодовое слово, используемое для обозначения строки из более чем одного символа;
- N_7 — максимальная длина строки, которая может быть закодирована.

Алгоритм сжатия реализует три основные действия:

- поиск совпадений строк и кодирование;
- добавление новых строк к словарю;
- удаление старых строк из словаря.

Алгоритм всегда ищет в словаре строку максимальной длины, совпадающую с входной строкой. Передатчик разбивает входной поток на строки, присутствующие в словаре, и преобразует каждую строку в соответствующее кодовое слово. Поскольку все односимвольные строки уже находятся в словаре, весь входной поток может быть разбит на строки, присутствующие в словаре. Приемник принимает поток кодовых слов и преобразует каждое кодовое слово в соответствующую символьную строку. Алгоритм всегда ищет новые строки, которые можно добавить к словарю, а также заменить ими старые строки, которые вряд ли встретятся в будущем. Процедура выглядит следующим образом:

1. Входные символы обрабатываются, чтобы получить совпадающие строки большей длины.
2. Если совпадающая строка имеет максимальную длину (N_7 символов), тогда алгоритм передает код для этой строки и переходит к шагу 1.
3. В противном случае к совпадающей строке добавляется следующий символ, а сама строка добавляется к словарю и ей назначается код. Однако поскольку эта строка еще отсутствует в словаре получателя, алгоритм передает код оригинальной строки и использует оставшийся символ, чтобы опять начать с шага 1.

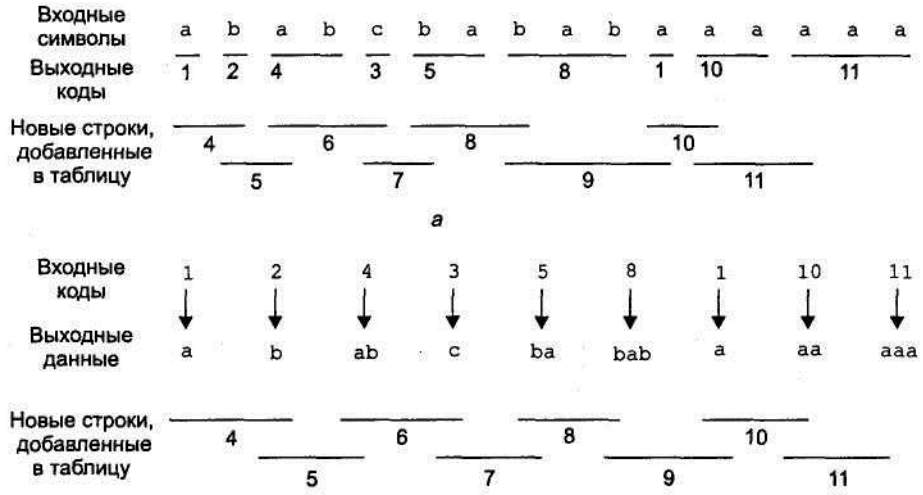
Процедура добавления новой строки к словарю зависит от того, является ли словарь полным или нет. В любом случае передатчик хранит переменную C_1 представляющую собой значение следующего доступного кодового слова. При инициализации системы переменной C_1 присваивается значение N_5 , которое представляет собой следующее значение, после того как всем односимвольным строкам присвоены значения. Таким образом, по умолчанию значение C_1 начинается с 256. До тех пор пока словарь пуст, при определении каждой новой строки ей назначается код C_1 и этот код увеличивается на единицу.

Когда словарь полон, при определении новой строки ей назначается кодовое значение C_1 затем выполняется следующая процедура:

1. C_1 увеличивается на единицу.
2. Если значение C_1 равно максимальному размеру словаря N_2, оно устанавливается равным N_5. То есть как только величина C_1 достигает своего максимального значения, она снова устанавливается равной минимальному значению.
3. Если узел, идентифицируемый значением C_1, не является листовым узлом, тогда выполняется переход к шагу 1.
4. Если узел является листовым узлом, тогда он удаляется из словаря.

В конце этой процедуры в словаре появляется место для новой записи, а C_1 представляет собой неиспользуемый код, присваиваемый этой записи. Теперь система готова к определению следующей новой строки словаря.

Рисунок ниже иллюстрирует работу алгоритма сжатия, для которого используется трехсимвольный алфавит. Вначале в словаре находятся только односимвольные строки (верхняя часть таблицы). Входные данные считываются слева направо. Поскольку в таблице нет совпадающих строк длиннее a, для этой строки выводится код 1, а к таблице добавляется новая строка ab, которой назначается код 4. Затем для начала новой строки используется символ b. Поскольку строки ba нет в словаре, она добавляется в словарь с кодом 5, а в выходной поток направляется символ b. Процесс продолжается подобным образом.

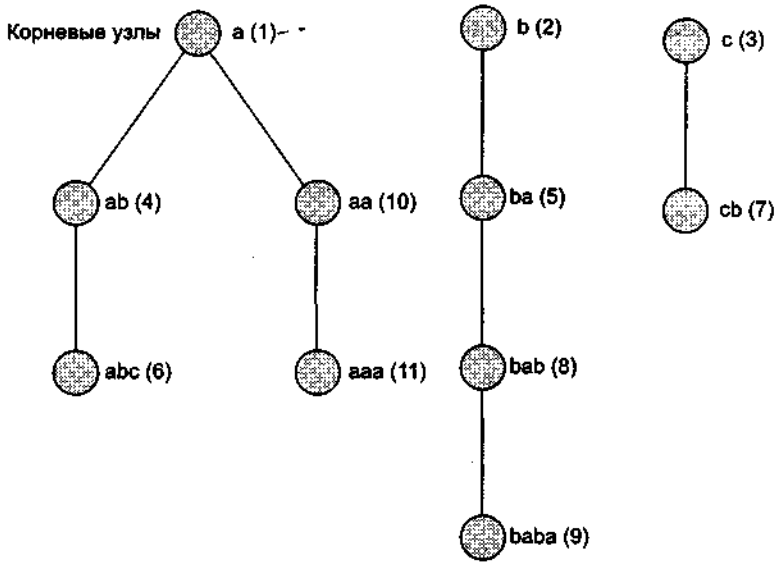


СТРОКОВАЯ ТАБЛИЦА		АЛЬТЕРНАТИВНАЯ ТАБЛИЦА	
СИМВОЛ	**КОД**	**СИМВОЛ**	**КОД**
A	**1**	**a**	**1**
B	**2**	**b**	**2**
c	**3**	**c**	**3**
ab	**4**	**1b**	**4**
ba	**5**	**2a**	**5**
abc	**6**	**4c**	**6**
cb	**7**	**3b**	**7**
bab	**8**	**5b**	**8**
baba	**9**	**8a**	**9**

aa	**10**	**1a**	**10**
Aaaa	**11**	**10a**	**11**

В таблице показан словарь, формируемый в данном примере. Более компактный метод хранения иллюстрирует правая часть таблицы, в которой каждая строка представляется в виде префиксного кода строки и последнего символа. При таком представлении все многосимвольные элементы словаря имеют равную длину.

Такое представление также предполагает структуру словаря, состоящую из нескольких деревьев, как показано на рисунке.



Распаковка

Интересный аспект работы семейства алгоритмов LZ78 заключается в том, что словарь не передается явно алгоритму распаковки. Вместо этого алгоритм распаковки создает идентичный словарь в процессе распаковки. Это возможно благодаря тому, что при распаковке строка, соответствующая коду, всегда встречается прежде самого кода.

Нижняя часть рисунка, описывающего процесс кодирования иллюстрирует процесс распаковки. Каждый встреченный код преобразуется в соответствующую символьную строку. Между тем, выходной поток используется для создания новых элементов словаря по тем же правилам, что и в алгоритме сжатия.

ЗАДАНИЕ

Построить код LZ77 для произвольной последовательности из нулей и единиц длинной 32 символа, используя скользящий буфер предистории и упреждающий буфер с длинами по 8 символов каждый.

Лекция 10. Сжатие с потерями

Сжатие данных без потерь накладывает жесткую нижнюю границу на размер любого файла или сообщения. Этой границей является энтропия этого файла. Сжатие данных без потерь позволяет восстановить исходные несжатые данные с точностью до бита. Эта характеристика, как правило, необходима для текстовых файлов, баз данных, двоичных объектных файлов и т. д. Однако существует много типов файлов, для которых не требуется абсолютно точного восстановления исходных данных. Примеры таких

файлов — аудио- и видеоклипы, а также неподвижные изображения. В этих случаях допускается некоторое количество ошибок при восстановлении исходных данных, и может применяться сжатие с потерями.

Ключевым вопросом сжатия с потерями является компромисс между коэффициентом сжатия и точностью восстановленных данных. В общем, должно быть ясно, что чем выше коэффициент сжатия для данного алгоритма сжатия, тем ниже точность восстановленных данных. Цель любого алгоритма сжатия с потерями заключается в достижении высоких коэффициентов сжатия за счет потери некоторых битов таким образом, чтобы восстановленные данные были достаточно близки к оригиналу.

Дискретное косинусное преобразование

Дискретное косинусное преобразование (Discrete Cosine Transform, DCT) представляет собой преобразование массива пикселов в массив значений пространственной частоты. Это преобразование обратимо с точностью до ошибок округления. Дискретное косинусное преобразование не производит сжатия, но преобразует информацию об изображении в форму, более удобную для сжатия.

Одномерное дискретное косинусное преобразование

Предположим, что у нас есть одномерное изображение, представляющее собой линейный массив из N пикселов, каждый из которых обозначается некоторым значением яркости *p(x)* (0 < *x* < *N).* Таким образом, *p(x)* представляет собой пространственную функцию (а не функцию времени). Это изображение можно представить в виде суммы пространственно-частотных компонентов с частотами, изменяющимися от 0 до *N*-1:

$$p(x) = \sqrt{\frac{2}{N}} \sum_{f=0}^{N-1} C(f)S(f)\cos\left[\frac{(2x+1)\pi f}{2N}\right] =$$

$$= \frac{S(0)}{\sqrt{N}} + \sqrt{\frac{2}{N}} \sum_{f=1}^{N-1} S(f)\cos\left[\frac{(2x+1)\pi f}{2N}\right]. \qquad (10.1)$$

Здесь:

$$C(f) = \begin{cases} 1/\sqrt{2}, & f = 0, \\ 1, & f > 0. \end{cases}$$

Это напоминает представление непрерывной функции в виде ряда Фурье. В данном случае мы представляем дискретную функцию *p(x),* а частотные компоненты определены только для дискретных значений частот.

Чтобы получить формулу (10.1), нужно вычислить коэффициенты *{S(f), 0<f<N}.* Обратите внимание на то, что первое слагаемое формулы (10.1) представляет собой компонент с нулевой частотой, то есть постоянную составляющую. Это слагаемое должно быть равно среднему значению *p(x).* Поэтому:

$$\frac{S(0)}{\sqrt{N}} = \frac{1}{N} \sum_{x=0}^{N-1} p(x),$$

$$S(0) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} p(x).$$

Общая формула для *S(f)* выглядит следующим образом:

$$S(f) = \sqrt{\frac{2}{N}} C(f) \sum_{x=0}^{N-1} p(x)\cos\left[\frac{(2x+1)\pi f}{2N}\right]. \qquad (10.2)$$

Формула (10.2) называется одномерным дискретным косинусным преобразованием функции *p(x),* а формула (10.1) представляет собой обратное дискретное косинусное преобразование функции *S(f).*

Пример

Рассмотрим пример, в котором блок из восьми пикселов одноцветного изображения подвергается дискретному косинусному преобразованию. Как правило, двумерное дискретное косинусное преобразование применяется к блоку размером 8x8 пикселов, так что данный пример представляет собой вполне реалистичную одномерную версию. В примере для обозначения яркости пикселов используются 8-битовые значения, так что 0 обозначает белый цвет, а 255 — черный. Это также типично для одноцветных изображений.

а

43	138	148	183	208	254	248	148

б

p(0)	p(1)	p(2)	p(3)	p(4)	p(5)	p(6)	p(7)
–85	10	20	55	80	126	120	20

в

S(0)	S(1)	S(2)	S(3)	S(4)	S(5)	S(6)	S(7)
122	–129	–95	26	–73	4	–31	–11

г

–85	10	20	55	80	126	120	20

д

Рис. 10.1.

На рис. 10.1 (а) показано исходное изображение, а на рис. 10.1 (*б)* — вектор значений яркости для этого изображения. Для удобства вычтем из каждого изображения 128 — такая операция называется *смещением уровня* (level shift) — так, чтобы диапазон всех возможных значений яркости был симметричен относительно 0 (от -128 до 127), в результате получим рис. 10.1 (в). Затем производится дискретное косинусное преобразование:

$$S(f) = \frac{1}{2} C(f) \sum_{x=0}^{7} p(x) \cos\left[\frac{(2x+1)\pi f}{16}\right].$$

Эти вычисления соответствуют значениям, показанным на рис. 10.1 (г). Чтобы проверить полученные значения, сосчитаем обратное дискретное косинусное преобразование:

$$p(x) = \frac{1}{2} \sum_{f=0}^{7} C(f) S(f) \cos\left[\frac{(2x+1)\pi f}{16}\right].$$

В результате мы получим значения, показанные на рис. 10.1 (*д),* которые точно совпадают с исходными значениями.

Стоит показать предыдущую формулу в развернутом виде, чтобы подчеркнуть природу дискретного косинусного преобразования:

$$p(x) = \frac{1}{\sqrt{8}} S(0) \cos[0] + \frac{S(1)}{2} \cos[(2x+1)\pi/16] +$$
$$+ \frac{S(2)}{2} \cos[(2x+1)2\pi/16] + \frac{S(3)}{2} \cos[(2x+1)3\pi/16] +$$
$$+ \frac{S(4)}{2} \cos[(2x+1)4\pi/16] + \frac{S(5)}{2} \cos[(2x+1)5\pi/16] +$$
$$+ \frac{S(6)}{2} \cos[(2x+1)6\pi/16] + \frac{S(7)}{2} \cos[(2x+1)7\pi/16].$$

Таким образом, функция *p(x)* может быть представлена в виде взвешенной суммы восьми косинусных функций. На рис. 10.2 показаны эти восемь функций, называемые базисными косинусными функциями. На каждом графике изображена косинусная функция дискретной пространственной переменной *x.* Каждая из этих дискретных функций представляет собой обычную косинусную функцию, дискретизированную по восьми отсчетам. Этим функциям присущи несколько важных свойств.

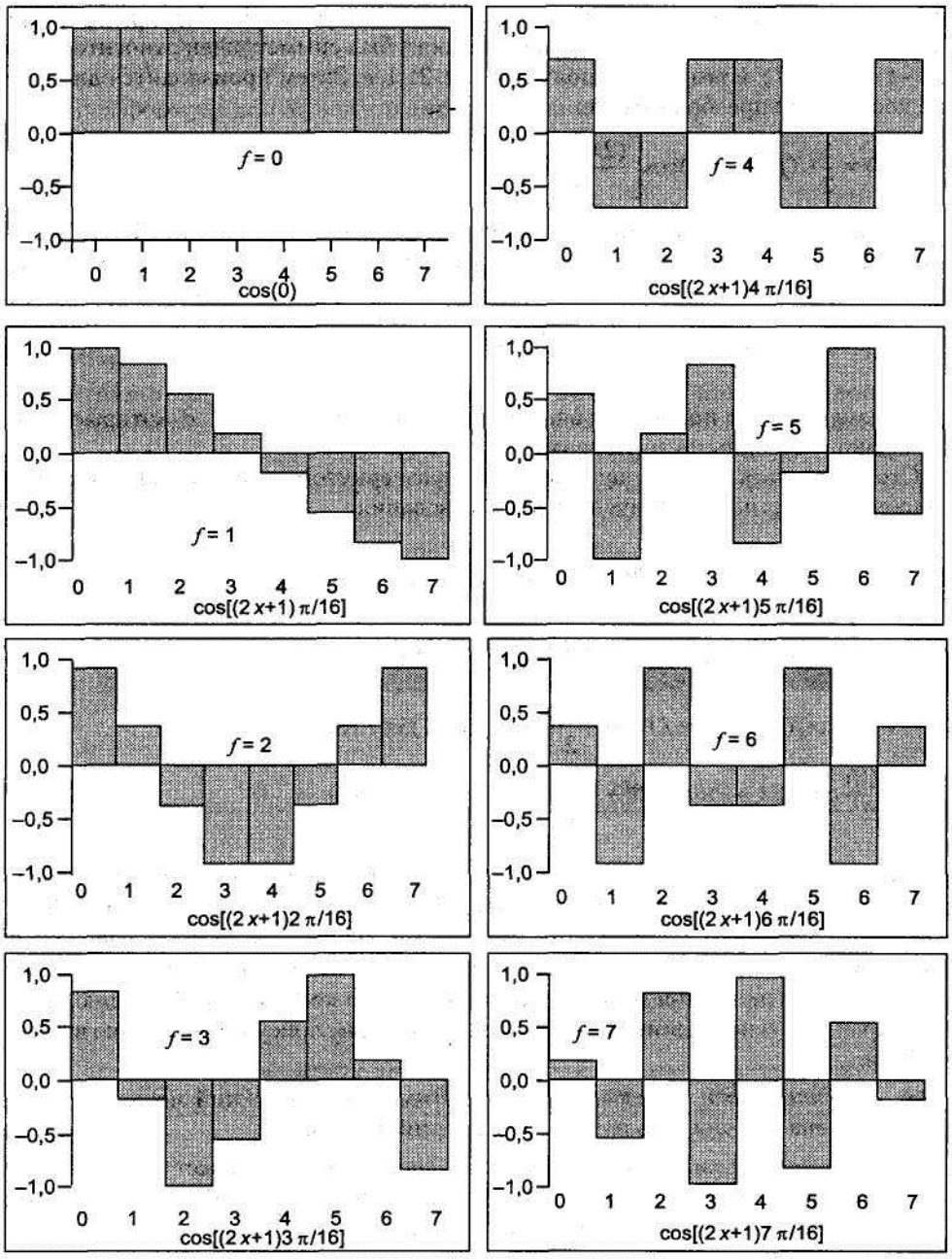


Рис. 10.2 Базисные дискретные косинусные функции

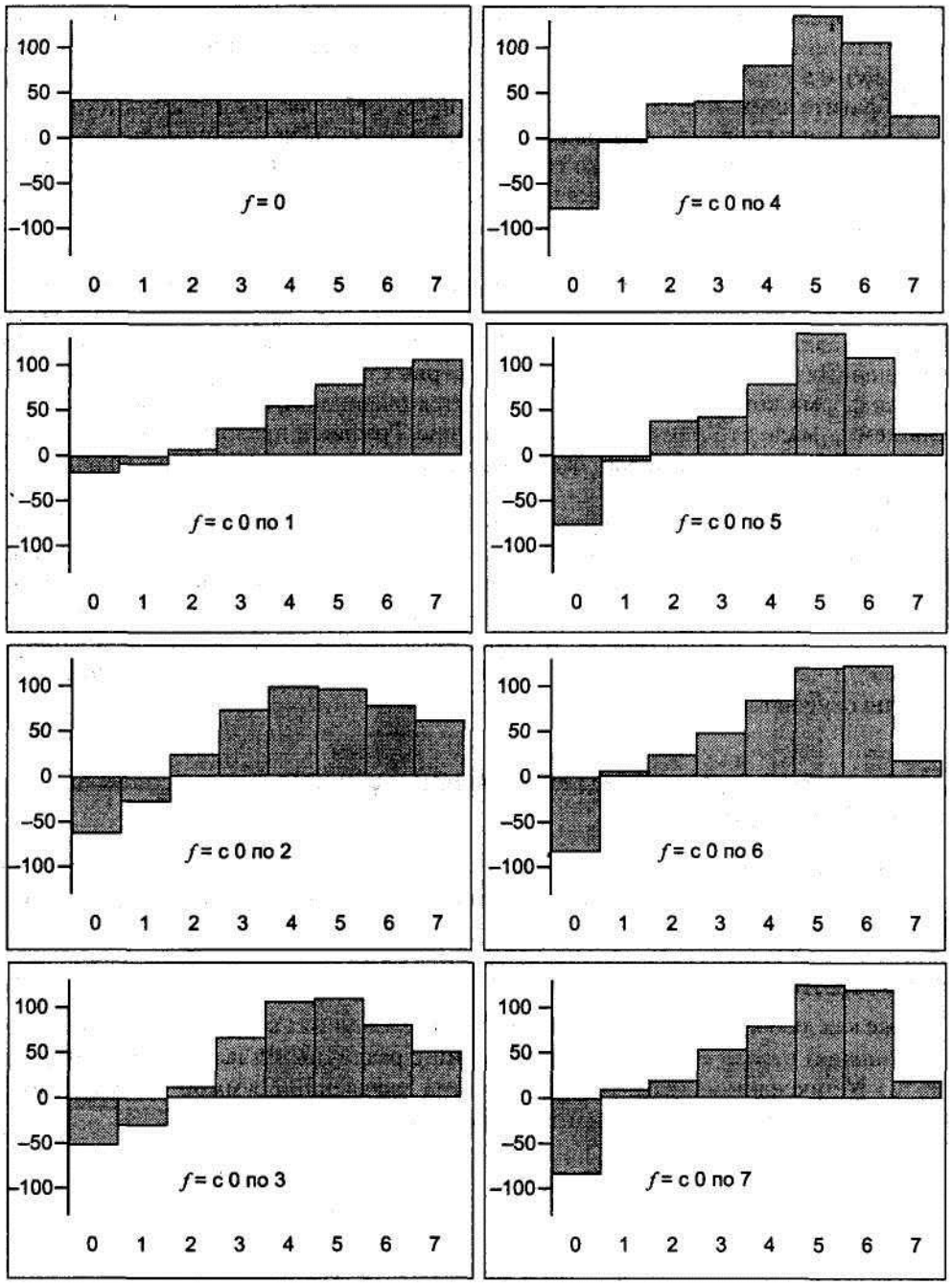


Рис. 10.3 Суммирование дискретных косинусных функций для примера на рис. 10.1

- *Завершенность.* Взвешенная сумма этих восьми функций может быть вычислена для каждого из восьми значений пикселов.
- *Минимальность.* Ни одна из восьми волновых форм не может быть представлена взвешенной комбинацией других волновых функций, и для завершенности требуются все восемь волновых форм.
- *Уникальность.* Никакой другой набор косинусных функций, кроме масштабированных версий этих восьми волновых форм, не может использоваться для представления всех возможных последовательностей восьмизначений пикселов.

Эти свойства также применимы к общему случаю использования *N* косинусных функций для представления одномерного изображения из N пикселов.

Двумерное дискретное косинусное преобразование

Двумерное дискретное косинусное преобразование является ключевым для стандартов JPEG и MPEG.

Так же как любой линейный массив из *N* пикселов может быть представлен в виде взвешенной суммы *N* косинусных функций с различными частотами, и матрица из *N x N* пикселов может быть представлена взвешенной суммой *N x N* косинусных функций. Формула выглядит следующим образом:

$$p(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)S(u,v)\cos\left[\frac{(2x+1)\pi u}{2N}\right]\cos\left[\frac{(2y+1)\pi v}{2N}\right]. \qquad (10.3)$$

Здесь

$$C(f)=\begin{cases}1/\sqrt{2}, & f=0,\\ 1, & f>0\end{cases}$$

для $f = u$ или v.

Как следует из формулы (10.3), двумерный массив пикселов может быть представлен произведениями пространственно-частотных компонентов в горизонтальном и вертикальном измерениях. Отдельные косинусные сомножители те же самые, что и в одномерном случае. Таким образом, для дискретного косинусного преобразования 8 x 8 используются те же косинусные функции, что и на рис. 10.3. Как и при одномерном преобразовании, слагаемое с нулевой частотой должно быть равно среднему значению функции *p(x)*. В данном случае это — *S(0, 0)/N*. Поэтому

$$\frac{S(0,0)}{N}=\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}p(x,y),$$

$$S(0,0)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}p(x,y).$$

Общая формула для $S(f)$ выглядит следующим образом:

$$S(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}p(x,y)\cos\left[\frac{(2x+1)\pi u}{2N}\right]\cos\left[\frac{(2y+1)\pi v}{2N}\right]. \qquad (10.4)$$

Формула (10.4) называется двумерным дискретным косинусным преобразованием функции *p(x, y)*.

Пример

Мы рассмотрим пример в котором квадратный массив из 64 пикселов (8 x 8) одноцветного изображения подвергается дискретному косинусному преобразованию. В стандарте JPEG изображения обрабатываются блоками по 8x8. Этот размер был выбран по двум причинам. Во-первых, сложность вычислений быстро растет с увеличением размера обрабатываемого блока, поэтому блоки больших размеров оказывают чрезмерное давление на вычислительные ресурсы. Во-вторых, исследования различных изображений показали, что ограничения обрабатываемой области этими размерами не приводят к существенным потерям точности.

На рис. 21.4, *а* показана матрица значений яркости после смещения уровня в диапазон от -128 до 127. Обратите внимание на то, что значения пикселов изменяются незначительно. Это типично для больших изображений: в любом блоке 8x8, скорее всего, изменения яркости будут небольшими. Затем производится преобразование:

$$S(u,v)=\frac{C(u)C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}p(x,y)\cos\left[\frac{(2x+1)\pi u}{16}\right]\cos\left[\frac{(2x+1)\pi v}{16}\right]. \qquad (10.5)$$

В результате данных вычислений получается матрица, показанная на рис. 10.4, *б*. Верхний левый элемент матрицы представляет собой значение 5(0, 0), соответствующее среднему значению матрицы, умноженное на 8:

$$S(0,0)=\frac{1}{8}\sum_{x=0}^{7}\sum_{y=0}^{7}p(x,y)=8\cdot\frac{\sum_{x=0}^{7}\sum_{y=0}^{7}p(x,y)}{64}.$$

79	75	79	82	82	86	94	94
76	78	76	82	83	86	85	94
72	75	67	78	80	78	74	82
74	76	75	75	86	80	81	79
73	70	75	67	78	78	79	85
69	63	68	69	75	78	82	80
76	76	71	71	67	79	80	83
72	77	78	69	75	75	78	78

а

619	–29	8	2	1	–3	0	1
22	–6	–4	0	7	0	–2	–3
11	0	5	–4	–3	4	0	–3
2	–10	5	0	0	7	3	2
6	2	–1	–1	–3	0	0	8
1	2	1	2	0	2	–2	–2
–8	–2	–4	1	2	1	–1	1
–3	1	5	–2	1	–1	1	–3

б

Рис. 10.4 Пример дискретного косинусного преобразования

Значение этого элемента значительно превосходит значения всех остальных элементов, которые уменьшаются с ростом частоты (с удалением от левого верхнего элемента матрицы). Это типично для многих изображений. Другими словами, можно сказать, что большая часть информации в типичном изображении находится в области низкочастотных составляющих. Другими словами, как правило, у изображений нет резких изменений яркости.

Для проверки этих значений можно осуществить обратное дискретное косинусное преобразование:

$$p(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)S(u,v)\cos\left[\frac{(2x+1)\pi u}{16}\right]\cos\left[\frac{(2y+1)\pi v}{16}\right]. \qquad (10.6)$$

В результате подобных преобразований мы получим исходные входные данные, показанные на рис. 10.4, *а.*

Вэйвлет – сжатие

В последние годы значительный интерес получило использование метода волнового сжатия (wavelet compression) изображений. Двумя заслуживающими внимания примерами использования данного метода являются система идентификации отпечатков пальцев ФБР и последняя версия широко применяемого стандарта JPEG —JPEG 2000. Привлекательность метода волнового сжатия заключается во впечатляющих коэффициентах сжатия при высоких скоростях. Метод волнового сжатия также обладает высокой гибкостью. Например, он позволяет представлять различные области изображения с различными степенями разрешения и точности. Кроме того, метод волнового сжатия хорошо подходит для прогрессивной передачи изображения, при которой сначала передается нечеткая версия изображения, а более мелкие детали пересылаются в ходе передачи позднее.

Элементарная волна

Хороший способ познакомиться с понятием *элементарной волны* (wavelet) — сравнение с представлением функции в виде ряда Фурье. Любая периодическая функция одной переменной *g(x)* может быть представлена в виде ряда Фурье:

$$g(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty}[A_n \cos(2\pi n f_0 x) + B_n \sin(2\pi n f_0 x)].$$

Здесь f_0 представляет собой величину, обратную периоду функции *(f_0 = 1/T).* Частота f_0 называется *основной частотой* (fundamental frequency) или *основной гармоникой* (fundamental harmonic). Кратные f_0 частоты называют *гармониками* (harmonics). Таким образом, периодическая функция с периодом *T* может быть представлена как сумма синусоид с частотами, кратными частоте $f_0 = 1/T$. Как правило, разложение в ряд Фурье применяется к функции времени (то есть аргумент *x* соответствует времени), но эта операция также применима и к пространственной функции, что уместно при обработке изображений. По существу, ряды Фурье показывают, что любая периодическая функция может быть полностью описана суммой синусоид с частотами, кратными основной частоте. Говорят, что ряды Фурье представляют функцию в *области частот,* тогда как оригинальная функция определена во *временной области* или в *пространстве* в зависимости от того, относится аргумент *x* ко времени или к пространству.

Используемые при разложении в ряд Фурье коэффициенты вычисляются по следующим формулам:

$$A_0 = \frac{2}{T}\int_0^T g(x)dx,$$

$$A_n = \frac{2}{T}\int_0^T g(x)\cos(2\pi n f_0 x)dx,$$

$$B_n = \frac{2}{T}\int_0^T g(x)\sin(2\pi n f_0 x)dx.$$

Таким образом, по представлению функции в виде ряда Фурье легко восстановить ее временную или пространственную форму. Непериодическая функция тоже может быть представлена в области частот при помощи преобразования Фурье. В этом случае вместо суммы дискретных гармоник функция представляется в непрерывном диапазоне частот в виде частотного спектра. Однако основные принципы здесь те же. Преобразование Фурье также обратимо, то есть можно по частотному спектру функции получить ее временное или пространственное представление.

Понятие представления Фурье можно распространить на двумерные функции и в таком виде использовать для обработки изображений.

Анализ Фурье представляет собой мощный инструмент, позволяющий решать многие проблемы, которые очень сложно решать другими методами. Коэффициенты Фурье можно вычислять, хранить, передавать и использовать для восстановления исходного сигнала или функции. При решении многих проблем значительно проще оперировать коэффициентами Фурье (в области частот), чем исходной функцией (во временной или пространственной области).

Однако анализ Фурье плохо подходит для работы с функциями, определенными на коротком интервале аргумента (в отличие от функций, заданных в диапазоне ($-\infty \le x \le \infty$), а также с функциями, значение которых резко и значительно изменяется. Такими характеристиками обладают изображения, имеющие ограниченную протяженность в пространстве и, как правило, содержащие резкие края или области с резко изменяющимися характеристиками. Для обработки таких изображений, а также для многих других целей анализ элементарных волн подходит лучше, чем анализ Фурье.

Как представление Фурье, волновое представление включает суммирование нескольких элементарных функций. В анализе Фурье в качестве элементарной функции используется синусоида, заданная на всей вещественной оси. В волновом анализе элементарной функцией является элементарная волна, представляющая собой функцию, отличную от нуля только на конечном отрезке значений аргумента и равную нулю для всех остальных значений аргумента. Функция представляется в виде суммы элементарных волн, каждая из которых, в свою очередь, представляет собой растянутую или суженную единичную элементарную волну, называемую *материнской элементарной волной* (mother wavelet). Путем анализа элементарных волн может быть осуществлено волновое преобразование сигнала, протяженного во времени, или изображения, в результате которого можно получить соответствующие коэффициенты. Как и в случае анализа Фурье, эти коэффициенты позволяют упростить обработку сигнала или изображения. Простейшая элементарная волна, применяемая в волновом анализе, называется элементарной волной Хаара (Haar wavelet) и определяется следующим образом:

$$\Psi_{j,k}(x) = \Psi_{0,0}(2^j x - k) = \begin{cases} 1, & k2^{-j} \le x < \left(k + \frac{1}{2}\right)2^{-j}, \\ -1, & \left(k + \frac{1}{2}\right)2^{-j} \le x < (k+1)2^{-j}, \\ 0, & \text{в остальных областях.} \end{cases}$$

Из этой базовой элементарной волны мы можем получить следующий набор функций:

На рис. 10.5 показаны некоторые из этих элементарных волн Хаара. Обратите внимание на то, что с увеличением параметра *j* элементарная волна становится более короткой, то есть частота элементарной волны растет. Параметр *k* сдвигает элементарную волну по оси *x* (во времени или в пространстве).

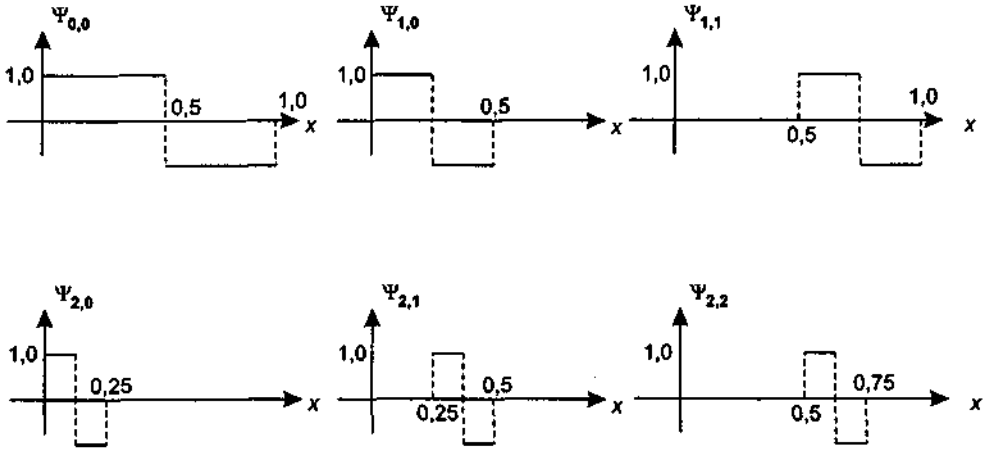


Рис. 10.5. Элементарные волны Хаара

Волновое представление позволяет использовать так называемый *анализ с переменным разрешением* (multiresolution analysis). Сжатые версии базовой элементарной волны, сдвинутые в соответствующие области сигнала или изображения, могут представлять высокочастотные составляющие. Несжатые версии базовой элементарной волны соответствуют низкочастотным, медленно изменяющимся компонентам. Добавляя все более сжатые элементарные волны, можно представлять сигнал или изображение с произвольной степенью разрешения или точности. Более того, подобное представление с переменным разрешением может найти очевидное применение для сжатия и прогрессивного преобразования:

- Изображение можно сжать, отбросив коэффициенты с относительно не большими значениями. В результате будут отброшены элементарные волны, вклад которых в формирование изображения невелик.
- Передачу изображения по сети можно начинать с низкочастотных компонентов. По мере приема более высокочастотных компонентов к изображению будет добавляться все больше деталей, а разрешение изображения будет увеличиваться, то есть изображение будет становиться четче.

Одномерное сжатие с помощью элементарных волн Хаара

Чтобы понять, как можно использовать элементарные волны Хаара для сжатия изображения в одном направлении, воспользуемся примером:

Рассмотрим строку из восьми значений данных, которые могут быть рядом из матрицы 8x8 пикселов, значение каждого из которых соответствует яркости. Их значения равны.

64 48 16 32 56 56 48 24

Мы будем преобразовывать изображение в три этапа, используя процесс, называемый усреднением и дифференцированием:

64 48 16 32 56 56 48 24
56 24 56 36 8 −8 0 12
40 46 16 10 8 −8 0 12
43 −3 16 10 8 −8 0 12

На первом шаге вычисляются средние значения последовательных пар чисел из исходной строки, в результате чего получаются первые четыре числа второй строки. Остальные четыре значения второй строки вычисляются как разности тех же пар чисел первого ряда. Таким образом, последние четыре числа второго ряда представляют собой уровень более мелкой детализации, отражающий изменения изображения. Первые четыре числа отражают фоновый уровень, к которому для воспроизведения оригинального изображения необходимо добавить более мелкие детали. На следующем шаге обрабатываются только первые четыре

числа, которые снова попарно усредняются и дифференцируются. В результате мы получаем два числа, представляющие уровень меньшей степени детализации. На последнем шаге усредняются и дифференцируются два первых значения предыдущего ряда. Все разностные значения показаны полужирным шрифтом. Обратите внимание на то, что первое значение последнего ряда представляет собой среднее значение всех восьми чисел исходного изображения. Мы можем сделать несколько наблюдений:

- Этот процесс обратим. Путем соответствующих сложений и вычитаний мы мжем снова получить из последнего ряда исходный ряд.
- Этот процесс можно обобщить применительно к строкам любой длины. Для троки из 2^k элементов требуются k этапов обработки. Любая строка, длина которой не равна степени 2, может быть дополнена нулями.
- Области, в которых исходные данные изменяются незначительно, после преобразования выглядят как последовательности небольших или нулевых значений.
- Преобразованное представление позволяет осуществить сжатие без потерь, так как обычно преобразованные данные содержат нулевые значения. В нашем примере седьмой элемент в преобразованном представлении равен нулю. Путем группового кодирования можно получить сжатую версию.
- Значительно большей степени сжатия, хотя и с потерями данных, можно добиться, игнорируя области малых изменений.

Чтобы продемонстрировать последнее утверждение, зададим некоторое пороговое значение ε и обнулим все элементы преобразованной строки, модуль которых не превышает ε. Например, при пороговом значении 4 из строки удаляется второй элемент, в результате чего получится строка (43, 0,16, 10, 8, -8, 0, 12). Поместим эту строку в последний ряд пустой таблицы и вычислим по ней исходные данные:

67	**51**	**19**	**35**	**53**	**53**	**45**	**21**
59	**27**	**53**	**33**	**8**	**–8**	**0**	**12**
40	**43**	**16**	**10**	**8**	**–8**	**0**	**12**
43	**0**	**16**	**10**	**8**	**–8**	**0**	**12**

На рис. 10.6 (а) полученный результат сравнивается с исходными данными. Соответствие оказывается довольно близким. На черно-белом дисплее эти два изображения были бы практически идентичными. Теперь установим пороговое значение ε = 8. При этом мы устраним еще два элемента в сжатой строке и получим строку (43,0,16,10, 0,0,0,12). Распакуем эту строку: (59,59, 27, 27, 53, 53,45, 21). Этот результат сравнивается с исходными данными на рис. 10.6 (*б).* Данное приближение основывается только на пяти из исходных восьми значений, но приближение остается довольно неплохим.

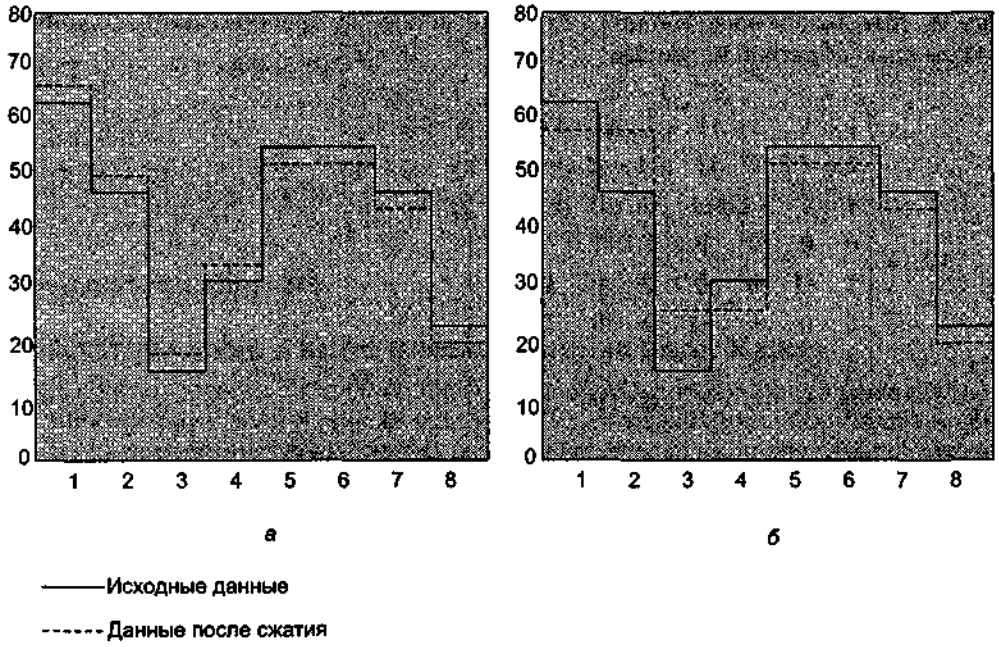


Рис. 10.6. Сжатие при помощи элементарных волн Хаара

Представление в виде матрицы

Для вычислений мы можем преобразовать только что описанные операции в простые конструкции линейной алгебры. Обратите внимание на то, что первый элемент в преобразованной строке представляет собой просто среднее значение всех точек исходных данных, то есть сумму всех значений, деленную на 8. Второй элемент представляет собой разность средних значений первых и последних четырех точек и т. д. Таким образом, мы можем представить данное преобразование в виде следующего произведения вектора и матрицы:

$$(64\ 48\ 16\ 32\ 56\ 48\ 24) \times \frac{1}{8} \times \begin{pmatrix} 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & -4 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 4 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & -4 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & -4 & 0 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & 4 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & -4 \end{pmatrix} =$$

$$= (43\ -3\ 16\ 10\ 8\ -8\ 0\ 12).$$

Определим два последних сомножителя в предыдущей формуле как матрицу W. Мы можем восстановить исходные данные из преобразованных данных с помощью обратной матрицы:

$$\mathbf{W}^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Убедиться в том, что матрица W^{-1} является обратной по отношению к W, можно, перемножив эти матрицы:

$$\mathbf{W}^{-1}\mathbf{W} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \times \frac{1}{8} \times$$

$$\times \begin{pmatrix} 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & -4 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 4 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & -4 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & -4 & 0 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & 4 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & -4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

С помощью обратной матрицы W^{-1} мы можем восстановить исходные данные из преобразованных данных:

$$(43\ -3\ 16\ 10\ 8\ -8\ 0\ 12) \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} =$$

$$= (64\ 48\ 16\ 32\ 56\ 48\ 24)$$

Представление в виде элементарных волн Хаара

Предыдущая формула предлагает способ выражения преобразования в виде суммы элементарных волн Хаара. Вспомним, что нам необходимо сформировать семейство элементарных волн, сжимая базовую элементарную волну в разное число раз и смещая ее по оси абсцисс. Таким образом, мы получим элементарную волну полного размера, две элементарные волны половинного размера, четыре элементарные волны размера 1/4 и т. д. С помощью подобных уменьшенных и сдвинутых элементарных волн мы можем представить ряды обратной матрицы W^{-1}, как показано на рис. 10.7.

Прежде чем продолжить, определим новую функцию, называемую масштабирующей функцией Хаара:

$$\varphi(x) = \begin{cases} 1, & \text{если } 0 \le x < 1, \\ 0 & \text{в противном случае.} \end{cases}$$

Теперь несложно выразить исходную строку данных в виде суммы элементарных волн Хаара и масштабирующей функции Хаара:

$$43\,\varphi(x) - 3\Psi_{0,0} + 16\Psi_{1,0} + 10\Psi_{1,1} + 8\Psi_{2,0} - 8\Psi_{2,1} + 0\Psi_{2,2} + 12\Psi_{2,3}.$$

Таким образом, мы можем интерпретировать строку как сумму общего среднего значения и семи элементарных волн, умноженных на соответствующие коэффициенты.

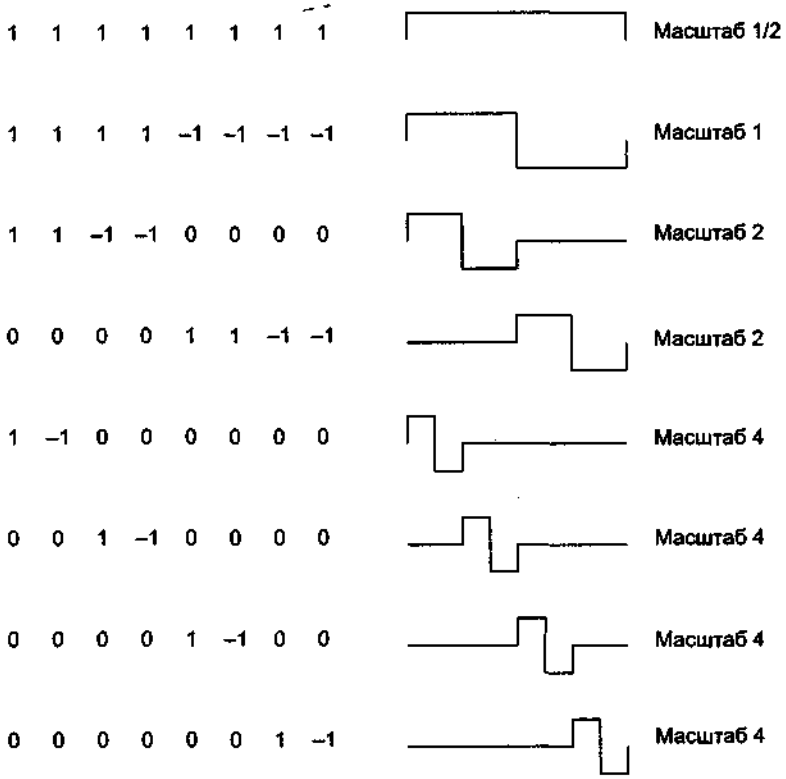


Рис. 10.7. Матрица, образованная элементарными волнами Хаара

Двумерное сжатие с помощью элементарных волн Хаара

Одномерное волновое преобразование Хаара заключается в обработке вектора-строки значений пикселов длины $N = 2^n$. Двумерное волновое преобразование Хаара включает обработку матрицы $N \times N$ значений пикселов. По существу, при двумерном волновом преобразовании Хаара сначала выполняются операции усреднения и дифференцирования с каждым рядом матрицы пикселов, а затем та же операция выполняется с каждым столбцом результата. Рассмотрим для примера матрицу пикселов 4 x 4:

$$\mathbf{P} = \begin{pmatrix} 18 & 14 & 12 & 4 \\ 10 & 6 & 8 & 8 \\ 16 & 4 & 8 & 0 \\ 12 & 0 & 4 & 4 \end{pmatrix}$$

Первый шаг двумерного волнового преобразования Хаара состоит из одномерного преобразования каждого ряда, осуществляемого путем умножения матриц PW, для чего используется версия 4x4 матрицы W, определенной в предыдущем подразделе:

$$\mathbf{PW} = \begin{pmatrix} 18 & 14 & 12 & 4 \\ 10 & 6 & 8 & 8 \\ 16 & 4 & 8 & 0 \\ 12 & 0 & 4 & 4 \end{pmatrix} \times \frac{1}{4} \times \begin{pmatrix} 1 & 1 & 2 & 0 \\ 1 & 1 & -2 & 0 \\ 1 & -1 & 0 & 2 \\ 1 & -1 & 0 & -2 \end{pmatrix} = \begin{pmatrix} 12 & 4 & 2 & 4 \\ 8 & 0 & 2 & 0 \\ 7 & 3 & 6 & 4 \\ 5 & 1 & 6 & 0 \end{pmatrix}$$

Следующий шаг состоит в выполнении одномерного преобразования каждого столбца. Для этого матрица преобразования столбцов транспонируется, затем умножается на преобразующую матрицу, после чего результат снова транспонируется. С тем же успехом можно транспонировать преобразующую матрицу и умножить ее на матрицу преобразованных столбцов:

$$\mathbf{T} = ((\mathbf{PW})'\mathbf{W})' = \mathbf{W}'\mathbf{PW}. \quad (10.7)$$

Здесь знак ' означает транспонирование. Таким образом, двумерное волновое преобразование исходной матрицы P равно

$$\mathbf{T} = \mathbf{W}'\mathbf{PW} = \frac{1}{4} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 2 & -2 & 0 & 0 \\ 0 & 0 & 2 & -2 \end{pmatrix} \begin{pmatrix} 12 & 4 & 2 & 4 \\ 8 & 0 & 2 & 0 \\ 7 & 3 & 6 & 4 \\ 5 & 1 & 6 & 0 \end{pmatrix} = \begin{pmatrix} 8 & 2 & 4 & 2 \\ 2 & 0 & -2 & 0 \\ 2 & 2 & 0 & 2 \\ 1 & 1 & 0 & 2 \end{pmatrix}$$

Преобразованная матрица T содержит среднее значение всех элементов исходной матрицы в левом верхнем углу (8), а остальные элементы соответствуют разностям. Элементы, отстоящие дальше от левого верхнего угла, соответствуют уровню более точной детализации, то есть более высокочастотным элементарным волнам. Как и раньше, сжатие может осуществляться при помощи порогового значения и удаления элементов, значение которых по модулю меньше порога.

Для обратного преобразования выполним следующие действия с формулой (10.7):

$$(\mathbf{W}')^{-1}\mathbf{T}\mathbf{W}^{-1} = (\mathbf{W}')^{-1}\mathbf{W}'\mathbf{PW}\mathbf{W}^{-1} = \mathbf{P}. \quad (10.8)$$

Две нужные нам матрицы легко определить:

$$\mathbf{W}^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \quad (\mathbf{W}')^{-1} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 0 & -1 & 1 & -1 \end{pmatrix}$$

Таким образом,

$$\mathbf{P}=\begin{pmatrix}1&1&1&0\\1&1&-1&0\\1&-1&0&1\\1&-1&0&-1\end{pmatrix}\begin{pmatrix}8&2&4&2\\2&0&-2&0\\2&2&0&2\\1&1&0&2\end{pmatrix}\begin{pmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{pmatrix}=$$

$$=\begin{pmatrix}1&1&1&0\\1&1&-1&0\\1&-1&0&1\\1&-1&0&-1\end{pmatrix}\begin{pmatrix}14&6&8&4\\0&4&2&2\\4&4&2&-2\\2&2&2&-2\end{pmatrix}=\begin{pmatrix}18&14&12&4\\10&6&8&8\\16&4&8&0\\12&0&4&4\end{pmatrix}$$