Компьютерные сети. Презентация для онлайн-обучения. Урок 1

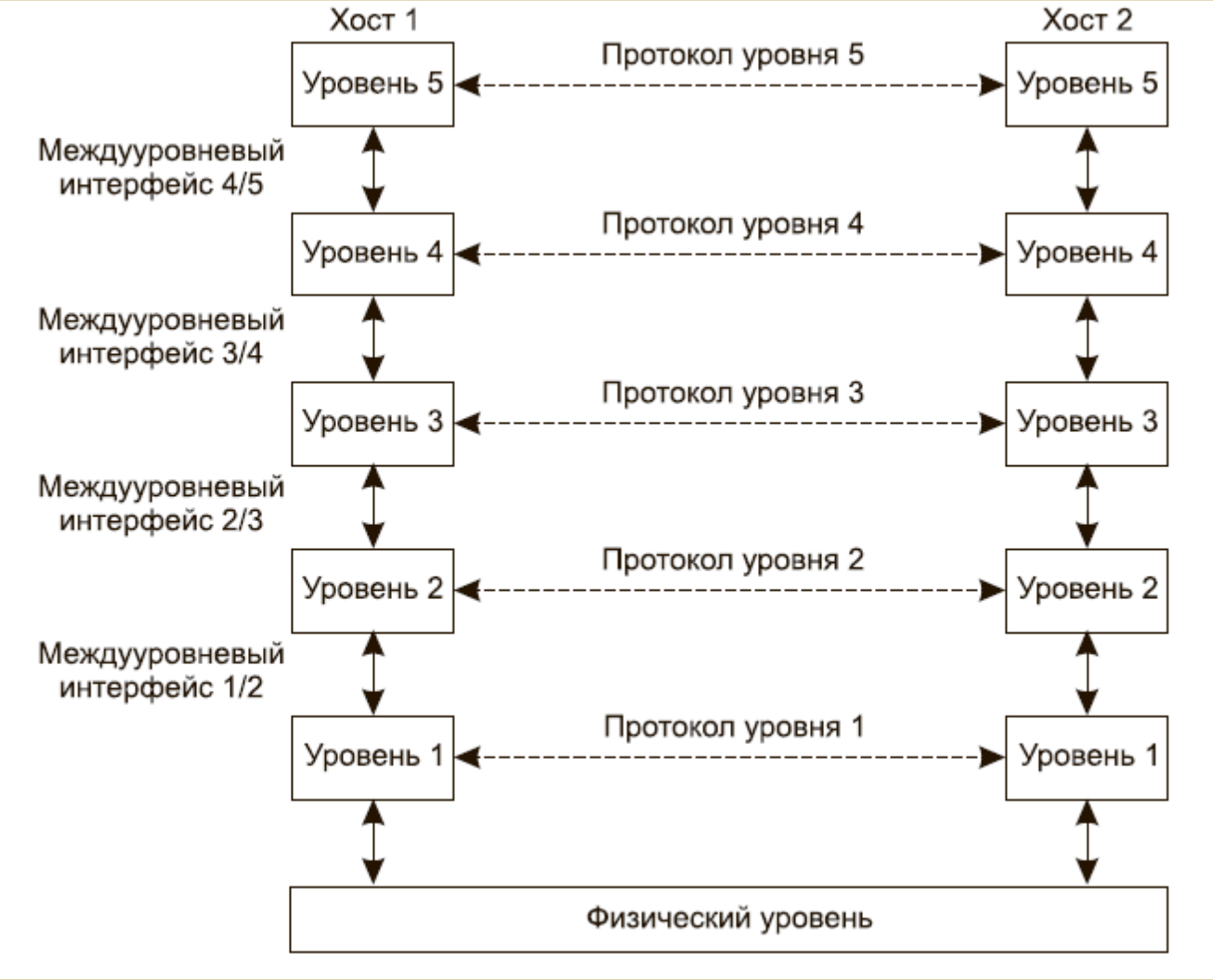
ИЕРАРХИЯ ПРОТОКОЛОВ

Для упрощения структуры большинство сетей организуются в наборы уровней или слоев, каждый последующий возводится над предыдущим. Количество уровней, их названия, содержание и назначение разнятся от сети к сети. Однако во всех сетях целью каждого уровня является предоставление неких сервисов для вышестоящих уровней. При этом от них скрываются детали реализации предоставляемого сервиса.

Такая концепция не нова и используется в вычислительной технике уже давно. Ее вариации известны как сокрытие информации, абстрактные типы данных, свойство инкапсуляции и объектно-ориентированное программирование. Фундаментальной идеей является предоставление неким программным или аппаратным уровнем сервисов своим пользователям без раскрытия деталей своего внутреннего состояния и подробностей алгоритмов.

Уровень n одной машины поддерживает связь с уровнем n другой машины. Правила и соглашения, используемые в данном общении, называются протоколом уровня n. По сути, протокол является договоренностью общающихся сторон о том, как должно происходить общение.

ИЕРАРХИЯ ПРОТОКОЛОВ



ИЕРАРХИЯ ПРОТОКОЛОВ

На рисунке показана пятиуровневая сеть. Объекты, включающие в себя соответствующие уровни на разных машинах, называются **равноранговыми** или равноправными узлами сети. Именно они общаются при помощи протокола. В действительности, данные не пересылаются с уровня n одной машины на уровень n другой машины. Вместо этого каждый уровень передает данные и управление уровню, лежащему ниже, пока не достигается самый нижний уровень. Ниже первого уровня располагается физическая среда, по которой и производится обмен информацией. На рисунке виртуальное общение показано пунктиром, тогда как физическое сплошными линиями.

Между каждой парой смежных уровней находится **интерфейс**, определяющий набор примитивных операций, предоставляемых нижним уровнем верхнему. Когда разработчики сетей решают, сколько уровней включить в сеть и что должен делать каждый уровень, одной из важнейших задач является определение ясных интерфейсов между уровнями. Подобная задача требует, в свою очередь, чтобы каждый уровень выполнял особый набор хорошо понятных функций.

ИЕРАРХИЯ ПРОТОКОЛОВ

В дополнение к минимизации количества информации, передаваемой между уровнями, ясно разграниченные интерфейсы также значительно упрощают изменение реализации уровня на совершенно другой протокол или реализацию (например, замену всех телефонных линий спутниковыми каналами), так как при этом всего лишь требуется, чтобы новый протокол или реализация предоставляла такой же набор услуг вышестоящему уровню, что и предыдущая. Вполне нормальное явление — использование хостов, принадлежащих к разным реализациям, одного и того же протокола (часто написанного различными компаниями). Фактически может изменяться сам протокол уровня, так что уровней выше и ниже это не затронет.

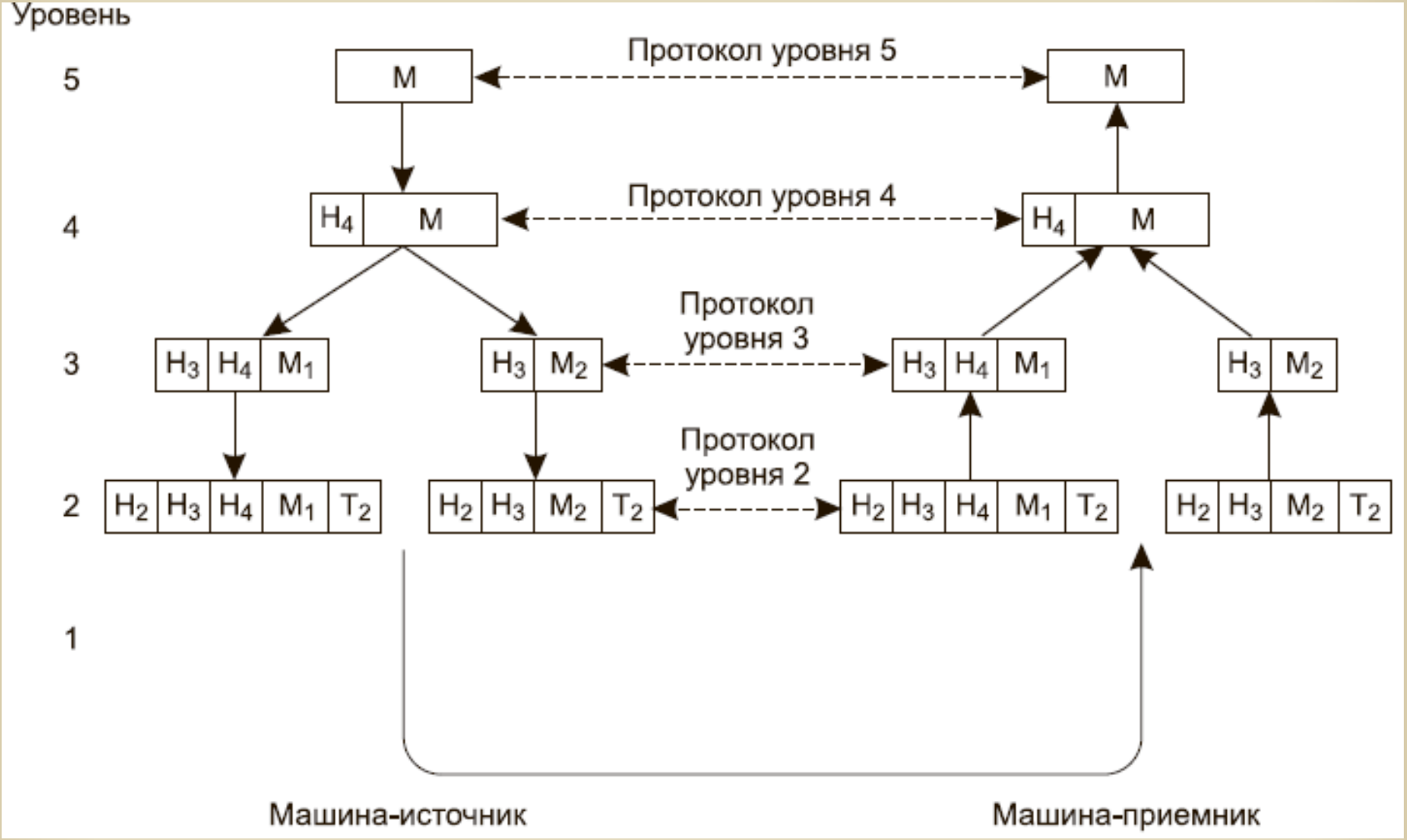
Набор уровней и протоколов называется архитектурой сети. Спецификация архитектуры должна содержать достаточно информации для написания программного обеспечения или создания аппаратуры для каждого уровня, чтобы они корректно выполняли требования протокола. Ни детали реализации, ни спецификации интерфейсов не являются частями архитектуры, так как они спрятаны внутри машины и не видны снаружи. При этом даже не требуется, чтобы интерфейсы на всех машинах сети были одинаковыми, лишь бы каждая машина правильно применяла все протоколы. Список протоколов, используемых системой, по одному протоколу на уровень, называется стеком протоколов.

ИЕРАРХИЯ ПРОТОКОЛОВ

В дополнение к минимизации количества информации, передаваемой между уровнями, ясно разграниченные интерфейсы также значительно упрощают изменение реализации уровня на совершенно другой протокол или реализацию (например, замену всех телефонных линий спутниковыми каналами), так как при этом всего лишь требуется, чтобы новый протокол или реализация предоставляла такой же набор услуг вышестоящему уровню, что и предыдущая. Вполне нормальное явление — использование хостов, принадлежащих к разным реализациям, одного и того же протокола (часто написанного различными компаниями). Фактически может изменяться сам протокол уровня, так что уровней выше и ниже это не затронет.

Набор уровней и протоколов называется архитектурой сети. Спецификация архитектуры должна содержать достаточно информации для написания программного обеспечения или создания аппаратуры для каждого уровня, чтобы они корректно выполняли требования протокола. Ни детали реализации, ни спецификации интерфейсов не являются частями архитектуры, так как они спрятаны внутри машины и не видны снаружи. При этом даже не требуется, чтобы интерфейсы на всех машинах сети были одинаковыми, лишь бы каждая машина правильно применяла все протоколы. Список протоколов, используемых системой, по одному протоколу на уровень, называется стеком протоколов.

ИЕРАРХИЯ ПРОТОКОЛОВ



ИЕРАРХИЯ ПРОТОКОЛОВ

Рассмотрим пример: как обеспечить общение для верхнего уровня пятиуровневой сети. Сообщение М производится приложением, работающим на уровне 5, и передается уровню 4 для передачи. Уровень 4 добавляет к сообщению заголовок для идентификации сообщения и передает результат уровню 3. Заголовок включает управляющую информацию, например адреса, позволяющие уровню 4 принимающей машины доставить сообщения. Другими примерами управляющей информации, используемой в некоторых уровнях, являются порядковые номера (в случае если нижний уровень не сохраняет порядок сообщения), размеры и время.

Во многих сетях сообщения, передаваемые на уровне 4, не ограничиваются по размеру, однако подобные ограничения почти всегда накладываются на протокол третьего уровня. Соответственно, уровень 3 должен разбить входящие сообщения на более мелкие единицы — пакеты, предваряя каждый пакет заголовком уровня 3. В данном примере сообщение М разбивается на две части, М1 и М2.

Уровень 3 решает, какую из выходных линий использовать, и передает пакеты уровню 2. Уровень 2 добавляет не только заголовки к каждому пакету, но также и завершающую последовательность с контрольной суммой (trailer), после чего передает результат уровню 1 для физической передачи. На получающей машине сообщение двигается по уровням вверх, при этом заголовки убираются на каждом уровне по мере продвижения сообщения. Заголовки нижних уровней более высоким уровням не передаются.

ИЕРАРХИЯ ПРОТОКОЛОВ

Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. Типичным примером **сервиса с установлением соединения** является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сначала устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов.

В некоторых случаях перед началом передачи отправляющая и получающая машины обмениваются приветствиями, отсылая друг другу приемлемые параметры соединения: максимальный размер сообщения, необходимое качество сервиса и др. В большинстве случаев одна из сторон посылает запрос, а другая его принимает, отвергает или же выставляет встречные условия. Линия — другое название соединения со связанными ресурсами, такими как фиксированная пропускная способность. Это название происходит из истории телефонной сети, в которой линия была путем по медному проводу, который переносил телефонный разговор.

ИЕРАРХИЯ ПРОТОКОЛОВ

Противоположный пример — сервисы без установления соединения, типичный пример такой технологии — почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некому маршруту, который совершенно не зависит от других писем. Есть различные названия для сообщений в различных контекстах; пакет — сообщение на сетевом уровне. Когда промежуточные узлы получают сообщение полностью перед пересылкой его к следующему узлу, это называют коммутацией с промежуточной буферизацией. Другой вариант, когда передача сообщения начинается прежде, чем оно будет полностью получено узлом, называют сквозной передачей. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше. Тем не менее возможна ситуация, что первое письмо задерживается и раньше приходит то, которое было послано вторым.

Каждая служба характеризуется качеством обслуживания. Некоторые службы являются надежными, в том смысле, что они никогда не теряют данные. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными.

.

ИЕРАРХИЯ ПРОТОКОЛОВ

Типичным примером необходимости надежной службы на основе соединений является пересылка файлов. Владелец файла хочет быть уверен, что все биты файла прибыли без искажений и в том же порядке, в котором были отправлены. Вряд ли кто-нибудь отдаст предпочтение службе, которая случайным образом искажает информацию, даже если передача происходит значительно быстрее.

Надежные службы на основе соединений бывают двух типов: последовательности сообщений и байтовые потоки. В первом варианте сохраняются границы между сообщениями. Когда посылаются два сообщения размером по 1 Кбайт, то они прибывают в виде двух сообщений размером по 1 Кбайт и никогда как одно двухкилобайтное сообщение. При втором варианте связь представляет собой просто поток байтов, без разделения на отдельные сообщения. Когда 2048 байт прибывают к получателю, то нет никакой возможности определить, было это одно сообщение длиной 2 Кбайт, два сообщения длиной 1 Кбайт или же 2048 однобайтных сообщений. Если страницы книги посылаются по сети фотонаборной машине в виде отдельных сообщений, то, возможно, необходимо сохранить границы между сообщениями. С другой стороны, чтобы загрузить DVD-фильм, вполне достаточно потока байтов с сервера на компьютер пользователя. Границы сообщений внутри фильма не важны.

ИЕРАРХИЯ ПРОТОКОЛОВ

Существуют системы, для которых задержки, связанные с пересылкой подтверждений, неприемлемы. В качестве примера такой системы можно назвать цифровую голосовую связь и IP-телефонию. В данном случае предпочтительнее допустить шумы на линии или искаженные слова, нежели большие паузы, вызванные отсылкой подтверждений и повторной передачей блоков данных. Аналогично, при проведении видеоконференции отдельные неправильные пикселы окажутся меньшей проблемой, нежели движение изображения резкими толчками; из-за того, что поток останавливается и начинает исправлять ошибки, мы видим дергающиеся и останавливающиеся кадры.

Не все приложения требуют установки соединения. Например, спаммеры рассылают рекламу по электронной почте большому количеству получателей. Спаммер, вероятно, не хочет устанавливать связь для пересылки каждого отдельного сообщения, а хочет отправить один объект. Также не требуется в этом случае и 100-процентная надежность, особенно если это существенно увеличит стоимость. Все, что нужно, — это способ переслать сообщение с высокой вероятностью его получения, но без гарантии. Ненадежная (то есть без подтверждений) служба без установления соединения часто называется **службой дейтаграмм** или дейтаграммной службой, по аналогии с телеграфной службой, также не предоставляющей подтверждений отправителю. Несмотря низкую надежность, это — доминирующая форма в большинстве сетей.

ИЕРАРХИЯ ПРОТОКОЛОВ

В других ситуациях бывает желательно не устанавливать соединение для пересылки сообщений, но надежность, тем не менее, существенна. Такая служба называется **службой дейтаграмм с подтверждениями**. Она подобна отправке заказного письма с подтверждением получения. Получив подтверждение, отправитель уверен, что письмо доставлено адресату, а не потеряно по дороге. Примером являются текстовые сообщения на мобильных телефонах.

Кроме того, существует служба запросов и ответов, в которой отправитель посылает дейтаграммы, содержащие запросы, и получает ответы от получателя. Обычно модель запросов и ответов применяется для реализации общения в модели клиент-сервер: клиент посылает запрос, а сервер отвечает на него. Например, пользователь мобильного телефона мог бы послать запрос в сервер карт, чтобы получить данные о карте для текущего местоположения.

Концепция использования ненадежной связи поначалу может показаться несколько странной. В самом деле, почему это может возникать такая ситуация, когда выгоднее предпочесть ненадежную связь надежной? Во-первых, надежное соединение (в том смысле, который был оговорен выше, то есть с подтверждением) не всегда можно установить на данном уровне. Скажем, Ethernet не является ≪надежным≫ средством коммуникации. Пакеты при передаче могут искажаться, но решать эту проблему должны протоколы более высоких уровней. В частности, много надежных служб создано над ненадежной службой дейтаграмм. Во-вторых, задержки, связанные с отсылкой подтверждения, в некоторых случаях неприемлемы, особенно при передаче мультимедиа в реальном времени. Именно благодаря этим факторам продолжают сосуществовать надежные и ненадежные соединения.

ИЕРАРХИЯ ПРОТОКОЛОВ

	Служба	Пример
Ориентированная на соединение	Надежный поток сообщений	Последовательность страниц
	Надежный поток байт	Удаленная регистрация
	Ненадежное соединение	Цифровая голосовая связь
Без установления соединения	Ненадежная дейтаграмма	Рассылка рекламы электронной почтой
	Дейтаграмма с подтверждениями	Заказные письма
	Запрос — ответ	Запрос к базе данных

ПРИМИТИВЫ СЛУЖБ

Служба (сервис) формально описывается набором примитивов или операций, доступных пользователю или другому объекту для получения сервиса. Эти примитивы заставляют службу выполнять некоторые действия или служат ответами на действия объекта того же уровня. Если набор протоколов входит в состав операционной системы (как часто и бывает), то примитивы являются системными вызовами. Они приводят к возникновению системных прерываний в привилегированном режиме, в результате чего управление машиной передается операционной системе, которая и отсылает нужные пакеты. Набор доступных примитивов зависит от природы сервиса. Скажем, примитивы сервисов с установлением соединения и без него различаются. В таблице приведен минимальный набор примитивов, обеспечивающий надежную передачу битового потока в среде типа «клиент-сервер». Примитивы являются аналогом сокет-интерфейса Беркли, поскольку примитивы — упрощенная версия этого интерфейса.

Примитив	Значение
LISTEN (ожидание)	Блокировка, ожидание входящего соединения
CONNECT (соединение)	Установка соединения с ожидающим объектом того же ранга
ACCEPT (прием)	Прием входящего соединения от объекта того же ранга
RECEIVE (прием)	Блокировка, ожидание входящего сообщения
SEND (отправка)	Отправка сообщения ожидающему объекту того же ранга
DISCONNECT (разрыв)	Разрыв соединения

ПРИМИТИВЫ СЛУЖБ

Эти примитивы могут использоваться для взаимодействия запрос-ответ в среде клиент-сервер. Чтобы проиллюстрировать, как это происходит, мы покажем набросок простого протокола, который осуществляет службу, используя общепризнанные дейтаграммы. Вначале сервер исполняет *LISTEN*, показывая тем самым, что он готов устанавливать входящие соединения. Этот примитив обычно реализуется в виде блокирующего системного вызова. После его исполнения процесс сервера приостанавливается до тех пор, пока не будет установлено соединение. Затем процесс клиента выполняет примитив *CONNECT*, устанавливая соединение с сервером. В системном вызове должно быть указано, с кем именно необходимо установить связь. Для этого может вводиться специальный параметр, содержащий адрес сервера. Далее операционная система клиента посылает равноранговой сущности пакет с запросом на соединение. Процесс клиента приостанавливается в ожидании ответа. Когда пакет приходит на сервер, операционная система обнаруживает запрос на соединение. Она проверяет, есть ли слушатель, и в этом случае разблокирует его. Затем процесс сервера может установить соединение с помощью *ACCEPT*. Он посылает ответ назад к процессу клиента, чтобы принять соединение. Прибытие этого ответа освобождает клиента. Начиная с этого момента считается, что сервер и клиент установили соединение.

ПРИМИТИВЫ СЛУЖБ

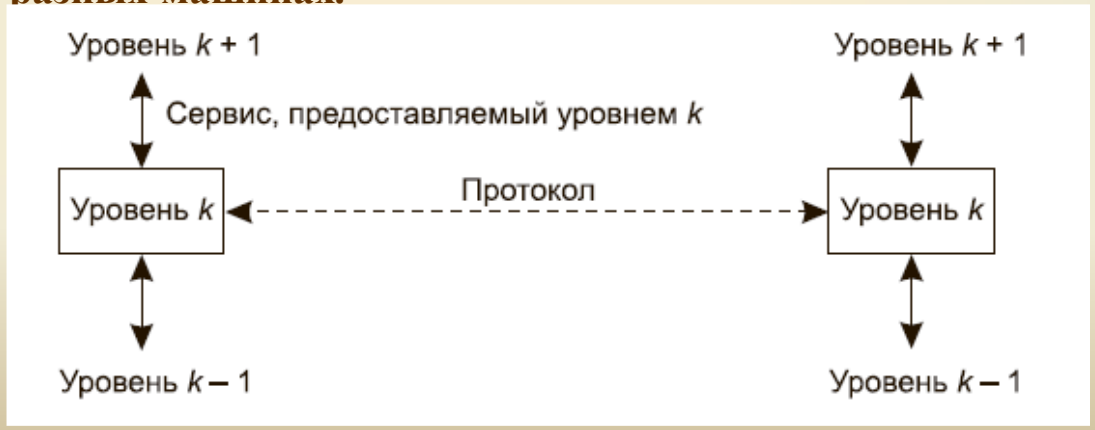
Следующим шагом будет выполнение сервером примитива *RECEIVE*, подготавливающего систему к принятию первого запроса. В нормальной ситуации это происходит сразу же после прекращения ожидания (*LISTEN*), даже до того, как клиент получает подтверждение соединения. Системный вызов *RECEIVE* вновь блокирует сервер. Клиент выполняет *SEND*, передает запрос (3) и сразу же выполняет *RECEIVE*, ожидая ответ. Прием пакета с запросом разблокирует сервер, благодаря чему он может обработать запрос. По окончании обработки сервер выполняет примитив *SEND*, и ответ отсылается клиенту (4). Прием пакета разблокирует клиента, теперь наступает его очередь обрабатывать пакет. Если у клиента есть еще запросы к серверу, он может отослать их. Когда запросы клиента окончены, он осуществляет разрыв соединения с помощью *DISCONNECT*. Обычно первый примитив *DISCONNECT* отсылает пакет, уведомляющий сервер об окончании сеанса, и блокирует клиента (5). В ответ сервер генерирует свой примитив *DISCONNECT*, являющийся подтверждением для клиента и командой, разрывающей связь. Клиент, получив его, разблокируется, и соединение считается окончательно разорванным. Именно так в двух словах можно описать схему коммуникации с установлением соединения.

СЛУЖБЫ И ПРОТОКОЛЫ

Службы и протоколы являются различными понятиями. Служба (или сервис)— это набор примитивов (операций), которые более низкий уровень предоставляет более высокому. Служба определяет, какие именно операции уровень будет выполнять от лица своих пользователей, но никак не оговаривает, как должны реализовываться эти операции. Служба описывает интерфейс между двумя уровнями, в котором нижний уровень является поставщиком сервиса, а верхний — его потребителем.

Напротив, протокол — это набор правил, описывающих формат и назначение кадров, пакетов или сообщений, которыми обмениваются объекты одного ранга внутри уровня. Объекты используют протокол для реализации определений своих служб. Они могут менять протокол по желанию, при условии, что при этом остаются неизменными службы, предоставляемые ими своим пользователям. Таким образом, служба и протокол оказываются практически независимыми. Это — ключевое понятие, которое должен хорошо понять любой проектировщик сетей.

Службы — это нечто, связанное с межуровневыми интерфейсами, тогда как протоколы связаны с пакетами, передающимися объектами одного уровня, расположенными на разных машинах.

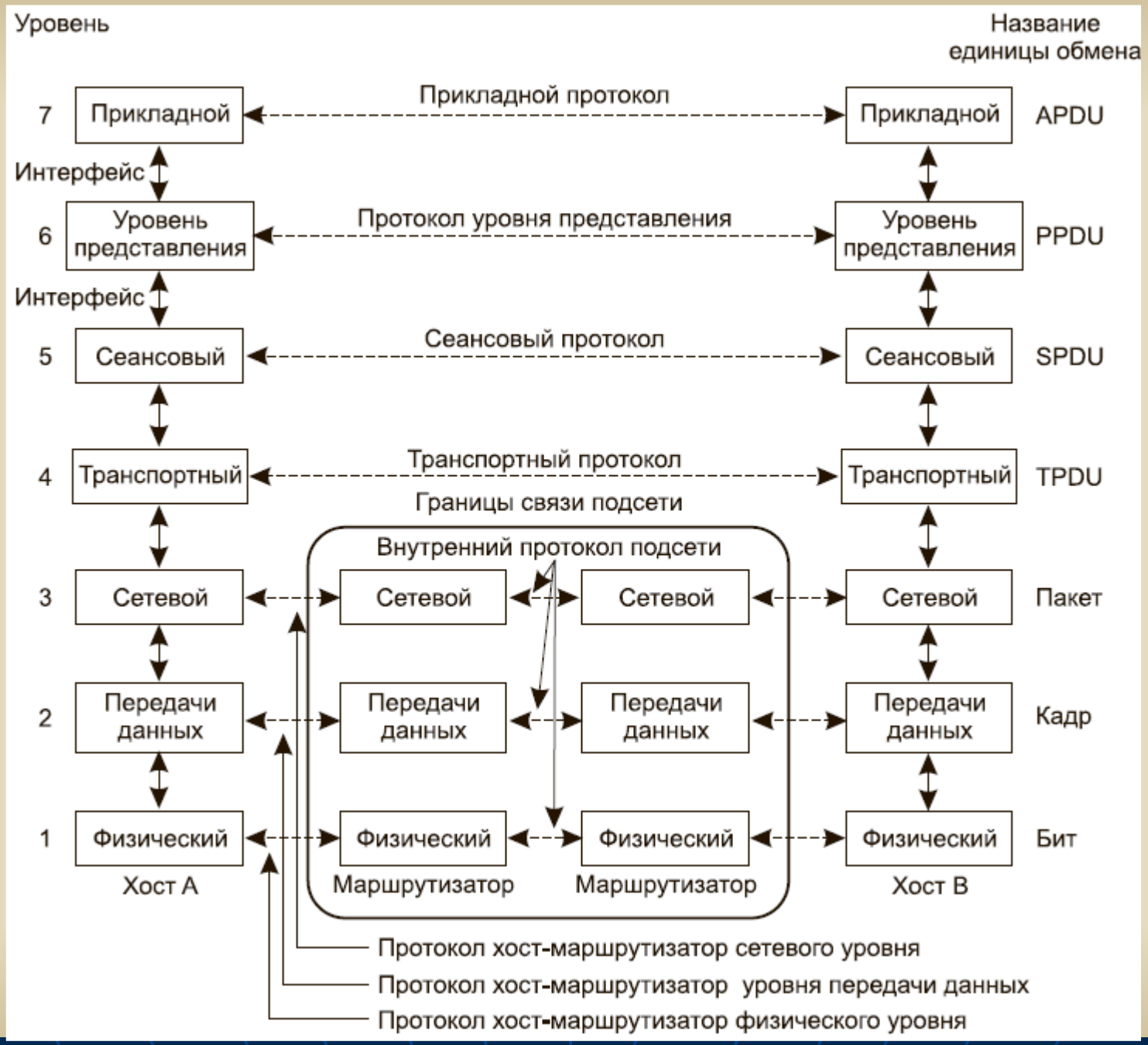


ЭТАЛОННАЯ МОДЕЛЬ OSI

Эта модель основана на разработке Международной организации по стандартизации (International Organization for Standardization, ISO) и является первым шагом к международной стандартизации протоколов, используемых на различных уровнях. Затем она была пересмотрена в 1995 году. Называется эта структура эталонной моделью взаимодействия открытых систем ISO, поскольку она связывает открытые системы, то есть системы, открытые для связи с другими системами. Для краткости мы будем называть эту модель просто ≪модель OSI≫. Модель OSI имеет семь уровней. Появление именно такой структуры было обусловлено следующими соображениями.

1. Уровень должен создаваться по мере необходимости отдельного уровня абстракции.
2. Каждый уровень должен выполнять строго определенную функцию.
3. Выбор функций для каждого уровня должен осуществляться с учетом создания стандартизированных международных протоколов.
4. Границы между уровнями должны выбираться так, чтобы поток данных между интерфейсами был минимальным.
5. Количество уровней должно быть достаточно большим, чтобы различные функции не объединялись в одном уровне без необходимости, но не слишком высоким, чтобы архитектура не становилась громоздкой.

ЭТАЛОННАЯ МОДЕЛЬ OSI



ФИЗИЧЕСКИЙ УРОВЕНЬ

Физический уровень занимается реальной передачей необработанных битов по каналу связи. При разработке сети необходимо убедиться, что когда одна сторона передает единицу, то принимающая сторона получает также единицу, а не ноль. Принципиальными вопросами здесь являются следующие: какое напряжение должно использоваться для отображения единицы, а какое для нуля; сколько микросекунд длится бит; может ли передача производиться одновременно в двух направлениях; как устанавливается начальная связь и как она прекращается, когда обе стороны закончили свои задачи; из какого количества проводов должен состоять кабель и какова функция каждого провода. Вопросы разработки в основном связаны с механическими, электрическими и процедурными интерфейсами, а также с физическим носителем, лежащим ниже физического уровня.

ТРАНСПОРТНЫЙ УРОВЕНЬ

Основная задача уровня передачи данных— быть способным передавать ≪сырые≫ данные физического уровня по надежной линии связи, свободной от необнаруженных ошибок, и маскировать реальные ошибки, так что сетевой уровень их не видит. Эта задача выполняется при помощи разбиения входных данных на кадры, обычный размер которых колеблется от нескольких сот до нескольких тысяч байт. Кадры данных передаются последовательно с обработкой кадров подтверждения, отсылаемых обратно получателем.

Еще одна проблема, возникающая на уровне передачи данных (а также и на большей части более высоких уровней), — как не допустить ситуации, когда быстрый передатчик заваливает приемник данными. Может быть предусмотрен некий механизм регуляции, который информировал бы передатчик о наличии свободного места в буфере приемника на текущий момент.

В широковещательных сетях существует еще одна проблема уровня передачи данных: как управлять доступом к совместно используемому каналу. Эта проблема разрешается введением специального дополнительного подуровня уровня передачи данных — подуровня доступа к носителю.

СЕТЕВОЙ УРОВЕНЬ

Сетевой уровень занимается управлением операциями подсети. Важнейшим моментом здесь является определение маршрутов пересылки пакетов от источника к пункту назначения. Маршруты могут быть жестко заданы в виде таблиц и редко меняться либо, что бывает чаще, автоматически изменяться, чтобы избегать отказавших компонентов. Кроме того, они могут задаваться в начале каждого соединения, например, терминальной сессии, такого как подключения к удаленной машине. Наконец, они могут быть в высокой степени динамическими, то есть вычисляемыми заново для каждого пакета с учетом текущей загруженности сети.

Если в подсети одновременно присутствует слишком большое количество пакетов, то они могут закрыть дорогу друг другу, образуя заторы в узких местах. Недопущение подобной закупорки также является задачей сетевого уровня в соединении с более высокими уровнями, которые адаптируют загрузку. В более общем смысле, сетевой уровень занимается предоставлением определенного уровня сервиса (это касается задержек, времени передачи, вопросов синхронизации).

СЕТЕВОЙ УРОВЕНЬ

При путешествии пакета из одной сети в другую также может возникнуть ряд проблем. Так, способ адресации, применяемый в одной сети, может отличаться от принятого в другой. Сеть может вообще отказаться принимать пакеты из-за того, что они слишком большого размера. Также могут различаться протоколы и т. д. Именно сетевой уровень должен разрешать все эти проблемы, позволяя объединять разнородные сети. В широковещательных сетях проблема маршрутизации очень проста, поэтому в них сетевой уровень очень примитивный или вообще отсутствует.

ТРАНСПОРТНЫЙ УРОВЕНЬ

Основная функция транспортного уровня — принять данные от сеансового уровня, разбить их при необходимости на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению. Кроме того, все это должно быть сделано эффективно и таким образом, чтобы изолировать более высокие уровни от каких-либо изменений в аппаратной технологии с течением времени.

Транспортный уровень также определяет тип сервиса, предоставляемого сеансовому уровню и, в конечном счете, пользователям сети. Наиболее популярной разновидностью транспортного соединения является защищенный от ошибок канал между двумя узлами, поставляющий сообщения или байты в том порядке, в каком они были отправлены. Однако транспортный уровень может предоставлять и другие типы сервисов, например пересылку отдельных сообщений без гарантии соблюдения порядка их доставки или одновременную отправку сообщения различным адресатам по принципу широковещания. Тип сервиса определяется при установке соединения. (Строго говоря, полностью защищенный от ошибок канал создать совершенно невозможно. Говорят лишь о таком канале, уровень ошибок в котором достаточно мал, чтобы им можно было пренебречь на практике.)

ТРАНСПОРТНЫЙ УРОВЕНЬ

Транспортный уровень является настоящим сквозным уровнем, то есть доставляющим сообщения от источника адресату. Другими словами, программа на машине-источнике поддерживает связь с подобной программой на другой машине при помощи заголовков сообщений и управляющих сообщений. На более низких уровнях для поддержки этого соединения устанавливаются соединения между всеми соседними машинами, через которые проходит маршрут сообщений.

СЕАНСОВЫЙ УРОВЕНЬ

Сеансовый уровень позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом. При этом предоставляются различные типы сервисов, среди которых управление диалогом (отслеживание очередности передачи данных), управление маркерами (предотвращение одновременного выполнения критичной операции несколькими системами) и синхронизация (установка служебных меток внутри длинных сообщений, позволяющих продолжить передачу с того места, на котором она оборвалась, даже после сбоя и восстановления).

УРОВЕНЬ ПРЕДСТАВЛЕНИЯ

В отличие от более низких уровней, задача которых — достоверная передача битов и байтов, уровень представления занимается по большей части синтаксисом и семантикой передаваемой информации. Чтобы было возможно общение компьютеров с различными внутренними представлениями данных, необходимо преобразовывать форматы данных друг в друга, передавая их по сети в неком стандартизированном виде. Уровень представления занимается этими преобразованиями, предоставляя возможность определения и изменения структур данных более высокого уровня (например, записей баз данных).

ПРИКЛАДНОЙ УРОВЕНЬ

Прикладной уровень содержит набор популярных протоколов, необходимых пользователям. Одним из наиболее распространенных является протокол передачи гипертекста HTTP (HyperText Transfer Protocol), который составляет основу технологии Всемирной паутины. Когда браузер запрашивает веб-страницу, он передает ее имя (адрес) и рассчитывает на то, что сервер, на котором расположена страница, будет использовать HTTP. Сервер в ответ отсылает страницу. Другие прикладные протоколы используются для передачи файлов, электронной почты, сетевых рассылок.

МОДЕЛЬ TCP/IP

Рассмотрим теперь эталонную модель, использовавшуюся в компьютерной сети ARPANET, а также в ее наследнице, всемирной сети Интернет. ARPANET была исследовательской сетью, финансируемой Министерством обороны США. В конце концов, она объединила сотни университетов и правительственных зданий при помощи выделенных телефонных линий. Когда впоследствии появились спутниковые сети и радиосети, возникли большие проблемы при объединении с ними других сетей с помощью имеющихся протоколов. Понадобилась новая эталонная архитектура. Таким образом, возможность объединять различные сети в единое целое являлась одной из главных целей с самого начала. Позднее эта архитектура получила название эталонной модели TCP/IP

Поскольку Министерство обороны США беспокоилось, что ценные хосты, маршрутизаторы и межсетевые шлюзы могут быть мгновенно уничтожены, другая важная задача состояла в том, чтобы добиться способности сети сохранять работоспособность при возможных потерях подсетевого оборудования, так чтобы при этом связь не прерывалась. Другими словами, Министерство обороны США требовало, чтобы соединение не прерывалось, пока функционируют приемная и передающая машины, даже если некоторые промежуточные машины или линии связи внезапно вышли из строя. Кроме того, от архитектуры нужна была определенная гибкость, поскольку предполагалось использовать приложения с различными требованиями, от переноса файлов до передачи речи в реальном времени.

КАНАЛЬНЫЙ УРОВЕНЬ

Все эти требования привели к выбору сети с пакетной коммутацией, основанной на уровне без установления соединения, который работает в различных сетях. Самый низкий уровень в модели, уровень канала, описывает то, как и что каналы, такие как последовательные линии и классический Ethernet, должны сделать, чтобы удовлетворить потребности этого межсетевого уровня без установления соединения. Это на самом деле не уровень вообще, в нормальном смысле слова, а скорее интерфейс между каналами передачи и узлами. В ранних материалах о модели TCP/IP мало что об этом говорится.

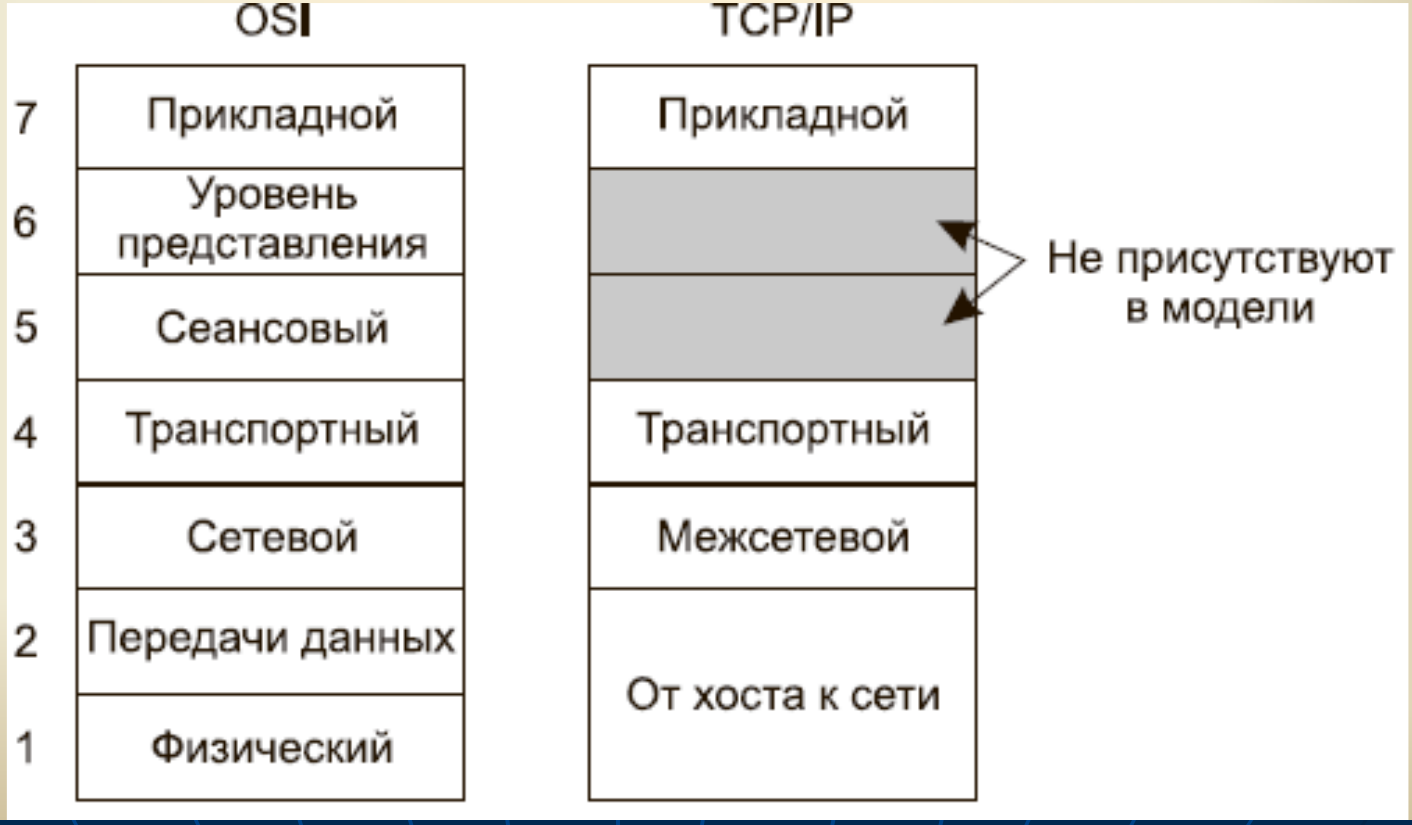
МЕЖСЕТЕВОЙ УРОВЕНЬ

Все эти требования обусловили выбор модели сети с коммутацией пакетов, в основе которой лежал не имеющий соединений межсетевой уровень. Он примерно соответствует сетевому уровню в OSI. Этот уровень, называемый интернет-уровнем или межсетевым уровнем, является основой всей архитектуры. Его задача заключается в обеспечении возможности каждого хоста посылать пакеты в любую сеть и независимо двигаться к пункту назначения (например, в другой сети). Они могут прибывать совершенно в другом порядке, чем были отправлены. Если требуется соблюдение порядка отправления, эту задачу выполняют более верхние уровни. Обратите внимание, что слово ≪интернет≫ здесь используется в своем первоначальном смысле, несмотря на то что этот уровень присутствует в сети Интернет.

Здесь можно увидеть аналогию с почтовой системой. Человек может бросить несколько международных писем в почтовый ящик в одной стране, и, если повезет, большая часть из них будет доставлена по правильным адресам в других странах. Вероятно, письма по дороге пройдут через несколько международных почтовых шлюзов, однако это останется тайной для корреспондентов. В каждой стране (то есть в каждой сети) могут быть свои марки, свои предпочитаемые размеры конвертов и правила доставки, незаметные для пользователей почтовой службы.

МЕЖСЕТЕВОЙ УРОВЕНЬ

Межсетевой уровень определяет официальный формат пакета и протокол IP, с дополнительным протоколом ICMP (Internet Control Message Protocol, межсетевой протокол управления сообщениями). Задачей межсетевого протокола является доставка IP-пакетов к пунктам назначения. Основными аспектами здесь являются выбор маршрута пакета и недопущение закупорки транспортных артерий (хотя IP не оказался эффективным для избегания скоплений).

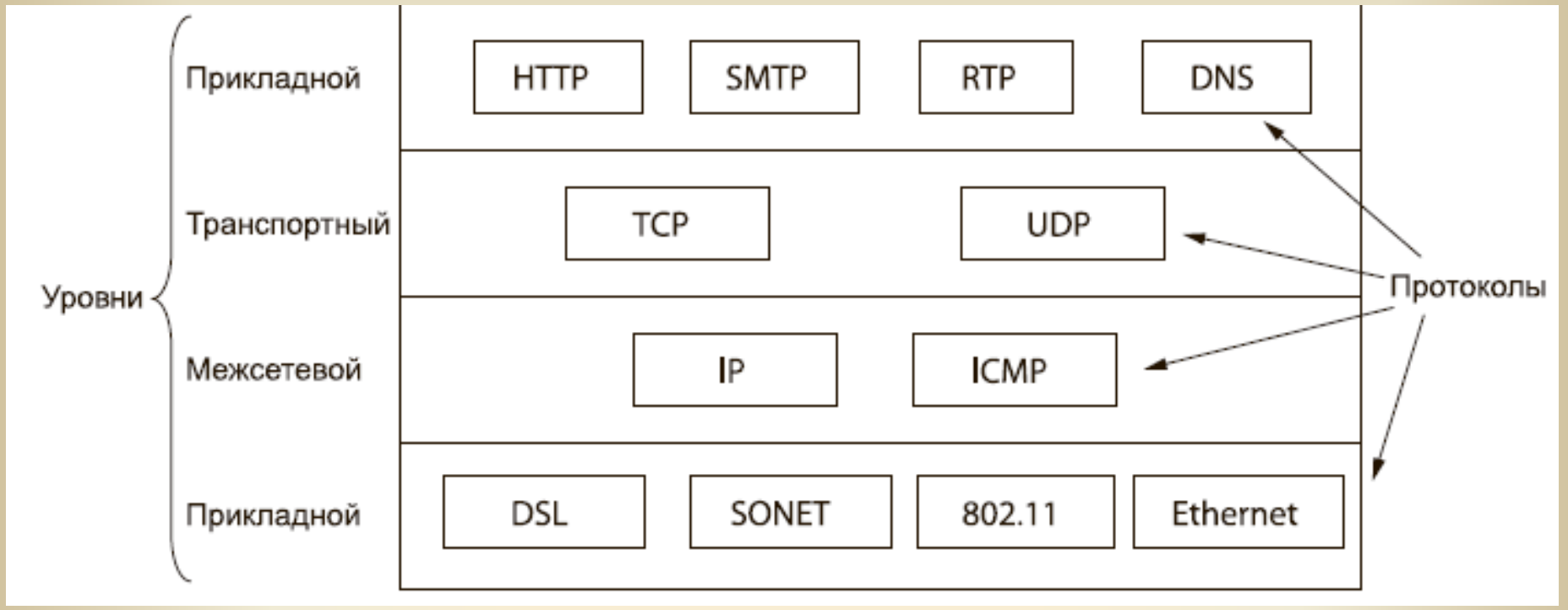


ТРАНСПОРТНЫЙ УРОВЕНЬ

Уровень, расположенный над межсетевым уровнем модели TCP/IP, как правило, называют транспортным. Он создан для того, чтобы объекты одного ранга на приемных и передающих хостах могли поддерживать связь, подобно транспортному уровню модели OSI. На этом уровне должны быть описаны два сквозных протокола. Первый, TCP (Transmission Control Protocol — протокол управления передачей), является надежным протоколом с установлением соединений, позволяющим без ошибок доставлять байтовый поток с одной машины на любую другую машину объединенной сети. Он разбивает входной поток байтов на отдельные сообщения и передает их межсетевому уровню. На пункте назначения получающий TCP-процесс собирает из полученных сообщений выходной поток. Кроме того, TCP осуществляет управление потоком, чтобы быстрый отправитель не завалил информацией медленного получателя.

Второй протокол этого уровня, UDP (User Dataram Protocol — протокол пользовательских дейтограмм), является ненадежным протоколом без установления соединения, не использующим последовательное управление потоком протокола TCP, а предоставляющим свое собственное. Он также широко используется в одноразовых клиент-серверных запросах и приложениях, в которых оперативность важнее аккуратности, например при передаче речи и видео. Со времени создания протокола IP этот протокол был реализован во многих других сетях.

ТРАНСПОРТНЫЙ УРОВЕНЬ



ПРИКЛАДНОЙ УРОВЕНЬ

В модели TCP/IP нет сеансового уровня и уровня представления. В этих уровнях про- сто не было необходимости, поэтому они не были включены в модель. Вместо этого приложения просто включают все функции сеансов и представления, которые им нужны. Опыт работы с моделью OSI доказал правоту этой точки зрения: большинство приложений мало нуждаются в этих уровнях.

Над транспортным уровнем располагается прикладной уровень. Он содержит все протоколы высокого уровня. К старым протоколам относятся протокол виртуального терминала (TELNET), протокол переноса файлов (FTP) и протокол электронной почты (SMTP). С годами было добавлено много других протоколов. Некоторые наиболее важные, мы рассмотрим. Это DNS (Domain Name Service — служба имен доменов), позволяющая преобразовывать имена хостов в сетевые, HTTP, протокол, используемый для создания страниц на World Wide Web, а также RTP, протокол для представления мультимедиа в реальном времени, таких как звук или фильмы.

СРАВНЕНИЕ МОДЕЛЕЙ OSI И TCP

У моделей OSI и TCP имеется много общих черт. Обе модели основаны на концепции стека независимых протоколов. Функциональность уровней также во многом схожа. Например, в обеих моделях уровни, начиная с транспортного и выше, предоставляют сквозную, не зависящую от сети транспортную службу для процессов, желающих обмениваться информацией. Эти уровни образуют поставщика транспорта. Также в каждой модели уровни выше транспортного являются прикладными потребителями транспортных сервисов.

Для модели OSI центральными являются три концепции.

1. Службы.

2. Интерфейсы.

3. Протоколы.

Вероятно, наибольшим вкладом модели OSI стало явное разделение этих трех концепций. Каждый уровень предоставляет некоторые сервисы для расположенного выше уровня. Сервис определяет, что именно делает уровень, но не то, как он это делает и каким образом объекты, расположенные выше, получают доступ к данному уровню. Интерфейс уровня определяет способ доступа к уровню для расположенных выше процессов. Он описывает параметры и ожидаемый результат. Он также ничего не сообщает о внутреннем устройстве уровня. Наконец, равноранговые протоколы, применяемые в уровне, являются внутренним делом самого уровня. Для выполнения поставленной ему задачи (то есть предоставления сервиса) он может использовать любые протоколы. Кроме того, уровень может менять протоколы, не затрагивая работу приложений более высоких уровней.

СРАВНЕНИЕ МОДЕЛЕЙ OSI И TCP

Эти идеи очень хорошо соответствуют современным идеям объектно-ориентированного программирования. Уровень может быть представлен в виде объекта, обладающего набором методов (операций), к которым может обращаться внешний процесс. Семантика этих методов определяет набор служб, предоставляемых объектом. Параметры и результаты методов образуют интерфейс объекта. Внутреннее устройство объекта можно сравнить с протоколом уровня. За пределами объекта оно никого не интересует и никому не видно.

Изначально в модели TCP/IP не было четкого разделения между службами, интерфейсом и протоколами, хотя и производились попытки изменить это, чтобы сделать ее более похожей на модель OSI. Так, например, единственными настоящими сервисами, предоставляемыми межсетевым уровнем, являются SEND IP PACKET (послать IPпакет) и RECEIVE IP PACKET (получить IP-пакет).

В результате в модели OSI протоколы скрыты лучше, чем в модели TCP/IP, и при изменении технологии они могут быть относительно легко заменены. Возможность проводить подобные изменения, не затрагивая другие уровни, является одной из главных целей многоуровневых протоколов.

СРАВНЕНИЕ МОДЕЛЕЙ OSI И TCP

Эталонная модель OSI была разработана прежде, чем были изобретены протоколы для нее. Такая последовательность событий означала, что эта модель не была настроена на какой-то конкретный набор протоколов, что делало ее универсальной. Обратной стороной такого порядка действий было то, что у разработчиков было мало опыта в данной области и не было четкого представления о том, какие функции должен выполнять каждый уровень.

Например, уровень передачи данных изначально работал только в сетях с передачей от узла к узлу. С появлением широковещательных сетей в модель потребовалось ввести новый подуровень. В дальнейшем, когда на базе модели OSI начали строить реальные сети с использованием существующих протоколов, обнаружилось, что они не соответствуют требуемым спецификациям служб. Поэтому в модель пришлось добавить подуровни для устранения несоответствия. Наконец, изначально ожидалось, что в каждой стране будет одна сеть, управляемая правительством и использующая протоколы OSI, поэтому никто и не думал об объединении различных сетей. В действительности все оказалось не так.

С моделью TCP/IP было все наоборот: сначала появились протоколы, а уже затем была создана модель, описывающая существующие протоколы. Таким образом, не было проблемы с соответствием протоколов модели. Они ей соответствовали прекрасно. Единственной проблемой было то, что модель не соответствовала никаким другим стекам протоколов. В результате она не использовалась для описания каких-нибудь других сетей, отличных от TCP/IP.

СРАВНЕНИЕ МОДЕЛЕЙ OSI И TCP

Если взглянуть на эти две модели поближе, то, прежде всего, обратит на себя внимание различие в количестве уровней: в модели OSI семь уровней, в модели TCP/ IP — четыре. В обеих моделях имеются межсетевой, транспортный и прикладной уровни, а остальные уровни различные.

Еще одно различие между моделями лежит в сфере возможности использования связи на основе соединений и связи без установления соединения. Модель OSI на сетевом уровне поддерживает оба типа связи, а на транспортном уровне — только связь на основе соединений (поскольку транспортные службы являются видимыми для пользователя). В модели TCP/IP на сетевом уровне есть только один режим связи (без установления соединения), но на транспортном уровне она поддерживает оба режима, предоставляя пользователям выбор. Этот выбор особенно важен для простых протоколов запрос-ответ.

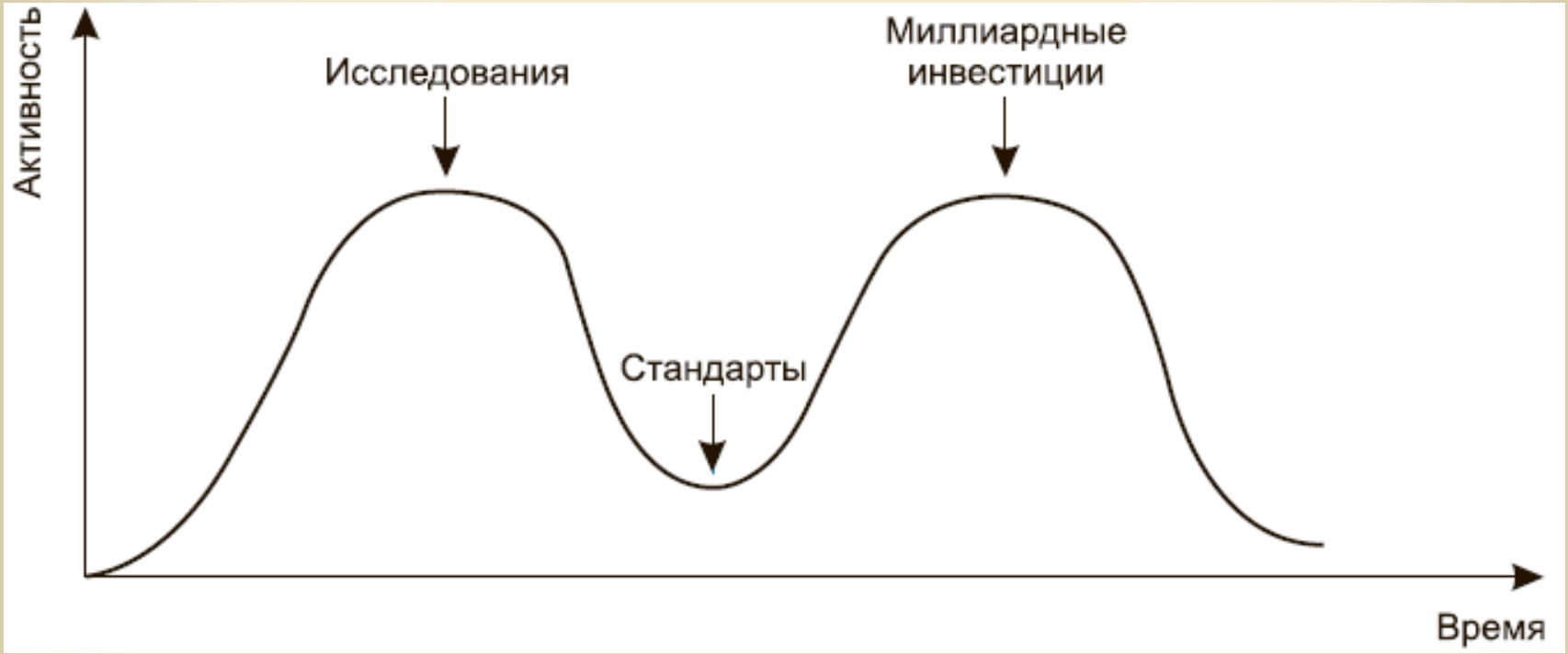
СРАВНЕНИЕ МОДЕЛЕЙ OSI И TCP

Если взглянуть на эти две модели поближе, то, прежде всего, обратит на себя внимание различие в количестве уровней: в модели OSI семь уровней, в модели TCP/ IP — четыре. В обеих моделях имеются межсетевой, транспортный и прикладной уровни, а остальные уровни различные.

Еще одно различие между моделями лежит в сфере возможности использования связи на основе соединений и связи без установления соединения. Модель OSI на сетевом уровне поддерживает оба типа связи, а на транспортном уровне — только связь на основе соединений (поскольку транспортные службы являются видимыми для пользователя). В модели TCP/IP на сетевом уровне есть только один режим связи (без установления соединения), но на транспортном уровне она поддерживает оба режима, предоставляя пользователям выбор. Этот выбор особенно важен для простых протоколов запрос-ответ.

КРИТИКА МОДЕЛИ OSI

Причина критики номер один: несвоевременность. Для успеха стандарта чрезвычайно важно, в какое время он устанавливается. У Дэвида Кларка (David Clark) из M.I.T. есть теория стандартов, которую он называет апокалипсисом двух слонов.



КРИТИКА МОДЕЛИ OSI

На рисунке изображена активность, сопровождающая любую новую разработку. Открытие новой темы вначале вызывает всплеск исследовательской активности в виде дискуссий, статей и собраний. Через некоторое время наступает спад активности, эту тему открывают для себя корпорации, и в результате в нее инвестируются миллиарды долларов.

Существенным является то, что стандарты пишутся именно в период между двумя «слонами». Если их создавать слишком рано, прежде чем закончатся исследования, предмет может оказаться еще слишком мало изучен и понят, что повлечет принятие плохих стандартов. Если создавать их слишком поздно, компании могут успеть вложить деньги в несколько отличные от стандартов технологии, так что принятые стандарты могут оказаться проигнорированными. Если интервал между двумя пика- ми активности будет слишком коротким (а все стремятся делать деньги как можно быстрее), разработчики стандартов могут просто не успеть их выработать.

Теперь становится ясно, почему стандартные протоколы OSI потерпели неудачу. К моменту их появления среди исследовательских университетов уже получили широкое распространение конкурирующие с ними протоколы TCP/IP. И хотя волна инвестиций еще не обрушилась на данную область, рынок университетов был достаточно широк для того, чтобы многие разработчики стали осторожно предлагать продукты, поддерживающие протоколы TCP/IP. Когда же появился OSI, разработчики не захотели поддерживать второй стек протоколов; таким образом, начальных предложений не было. Каждая компания выжидала, пока первым начнет кто-нибудь другой, поэтому OSI так никто и не стал поддерживать.

КРИТИКА МОДЕЛИ OSI

Второй причиной, по которой модель OSI не была реализована, оказалось несовершенство как самой модели, так и ее протоколов. Выбор семиуровневой структуры стал больше политическим решением, чем техническим. В результате два уровня (сеансовый и уровень представления) почти пусты, тогда как два других (сетевой и передачи данных) перегружены.

Эталонная модель OSI вместе с соответствующими определениями служб и протоколами оказалась невероятно сложной. Если сложить в стопку распечатку официального описания стандартов, получится кипа бумаги высотой в один метр. Модель тяжело реализуема и неэффективна в работе.

КРИТИКА МОДЕЛИ OSI

Учитывая огромную сложность модели и протоколов, громоздкость и медлительность первых реализаций не стали неожиданностью. Неудачу потерпели все, кто попытался реализовать эту модель. Поэтому вскоре понятие «OSI» стало ассоциироваться с плохим качеством. И хотя со временем продукты улучшились, ассоциации остались.

Первые реализации TCP/IP, основанные на Berkley UNIX, напротив, были достаточно хороши (не говоря уже о том, что они были открытыми). Они довольно быстро вошли в употребление, что привело к появлению большого сообщества пользователей. Это вызвало исправления и улучшения реализации, в результате чего сообщество пользователей еще выросло. В данном случае обратная связь явно была положительной.

КРИТИКА МОДЕЛИ TCP/IP

У модели TCP/IP и ее протоколов также имеется ряд недостатков. Во-первых, в этой модели нет четкого разграничения концепций служб, интерфейсов и протоколов. При разработке программного обеспечения желательно провести четкое разделение между спецификацией и реализацией, что весьма тщательно делает OSI и чего не делает TCP/IP. В результате модель TCP/IP довольно бесполезна при разработке сетей, использующих новые технологии.

Во-вторых, модель TCP/IP отнюдь не является общей и довольно плохо описывает любой стек протоколов, кроме TCP/IP. Так, например, описать технологию Bluetooth с помощью модели TCP/IP совершенно невозможно.

В-третьих, канальный уровень в действительности не является уровнем в том смысле, который обычно используется в контексте уровневых протоколов. Это скорее интерфейс между сетью и уровнями передачи данных. Различие между интерфейсом и уровнем является чрезвычайно важным, и здесь не следует быть небрежным.

В-четвертых, в модели TCP/IP не различаются физический уровень и уровень передачи данных. Об этом различии даже нет упоминания. Между тем, они абсолютно разные. Физический уровень должен иметь дело с характеристиками передачи информации по медному кабелю, оптическому волокну и по радио, тогда как задачей уровня передачи данных является определение начала и конца кадров и передача их с одной стороны на другую с требуемой степенью надежности. Правильная модель должна содержать их как два различных уровня. В модели TCP/IP этого нет.