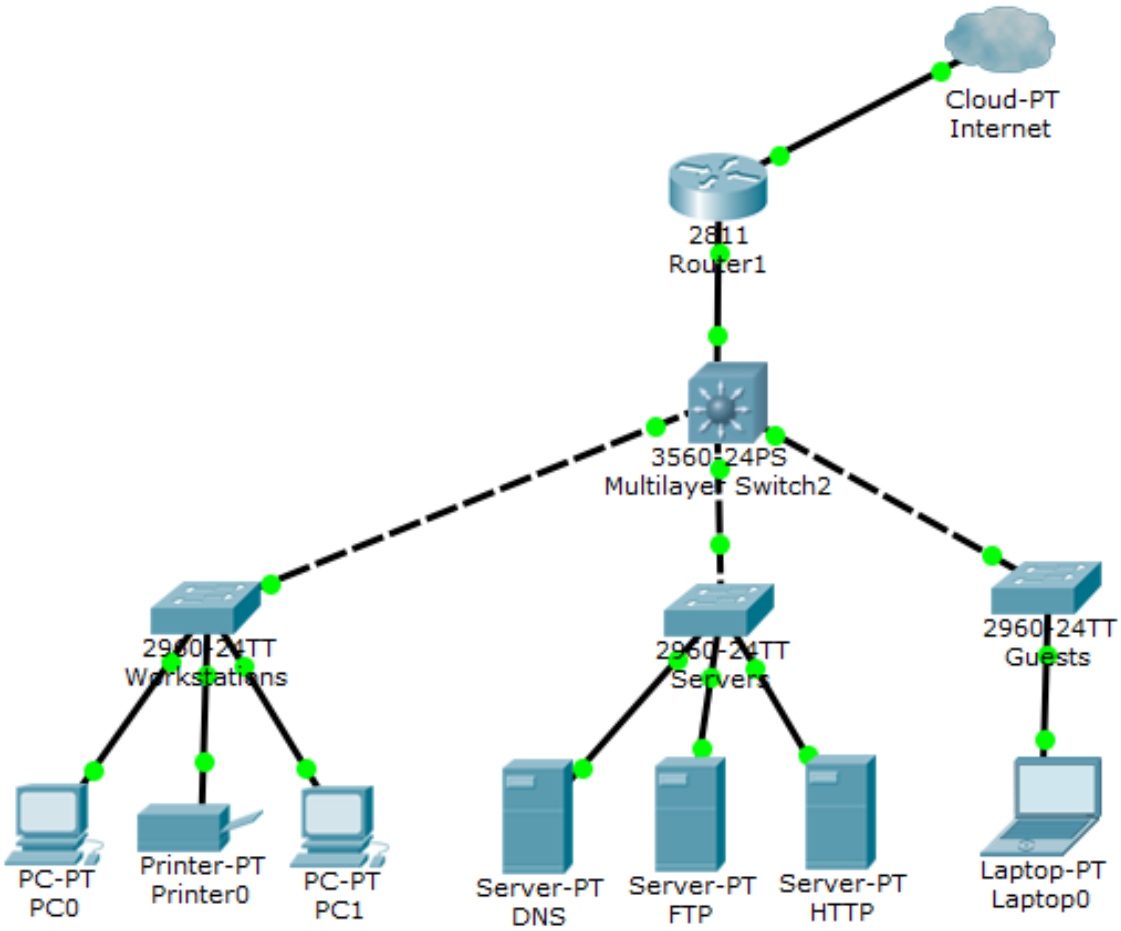
Основы компьютерных сетей. Тема №1. Основные сетевые термины и сетевые модели / Хабр

habr.com/ru/articles/307252

Денис 12 августа 2016 г.

Всем привет. На днях возникла идея написать статьи про основы компьютерных сетей, разобрать работу самых важных протоколов и как строятся сети простым языком. Заинтересовавшихся приглашаю под кат.



Немного оффтопа: Приблизительно месяц назад сдал экзамен CCNA (на 980/1000 баллов) и осталось много материала за год моей подготовки и обучения. Учился я сначала в академии Cisco около 7 месяцев, а оставшееся время вел конспекты по всем темам, которые были мною изучены. Также консультировал многих ребят в области сетевых технологий и заметил, что многие наступают на одни и те же грабли, в виде пробелов по каким-то ключевым темам. На днях пару ребят попросили меня объяснить, что такое сети и как с ними работать. В связи с этим решил максимально подробно и простым языком описать самые ключевые и важные вещи. Статьи будут полезны новичкам, которые только встали на путь изучения. Но, возможно, и бывалые сисадмины подчеркнут из этого что-то

полезное. Так как я буду идти по программе CCNA, это будет очень полезно тем людям, которые готовятся к сдаче. Можете держать статьи в виде шпаргалок и периодически их просматривать. Я во время обучения делал конспекты по книгам и периодически читал их, чтобы освежать знания.

Вообще хочу дать всем начинающим совет. Моей первой серьезной книгой, была книга Олиферов «Компьютерные сети». И мне было очень тяжело читать ее. Не скажу, что все было тяжело. Но моменты, где детально разбиралось, как работает MPLS или Ethernet операторского класса, вводило в ступор. Я читал одну главу по несколько часов и все равно многое оставалось загадкой. Если вы понимаете, что какие то термины никак не хотят лезть в голову, пропустите их и читайте дальше, но ни в коем случае не отбрасывайте книгу полностью. Это не роман или эпос, где важно читать по главам, чтобы понять сюжет. Пройдет время и то, что раньше было непонятным, в итоге станет ясно. Здесь прокачивается «книжный скилл». Каждая следующая книга, читается легче предыдущей книги. К примеру, после прочтения Олиферов «Компьютерные сети», читать Таненбаума «Компьютерные сети» легче в несколько раз и наоборот. Потому что новых понятий встречается меньше. Поэтому мой совет: не бойтесь читать книги. Ваши усилия в будущем принесут плоды. Заканчиваю разглагольствование и приступаю к написанию статьи.

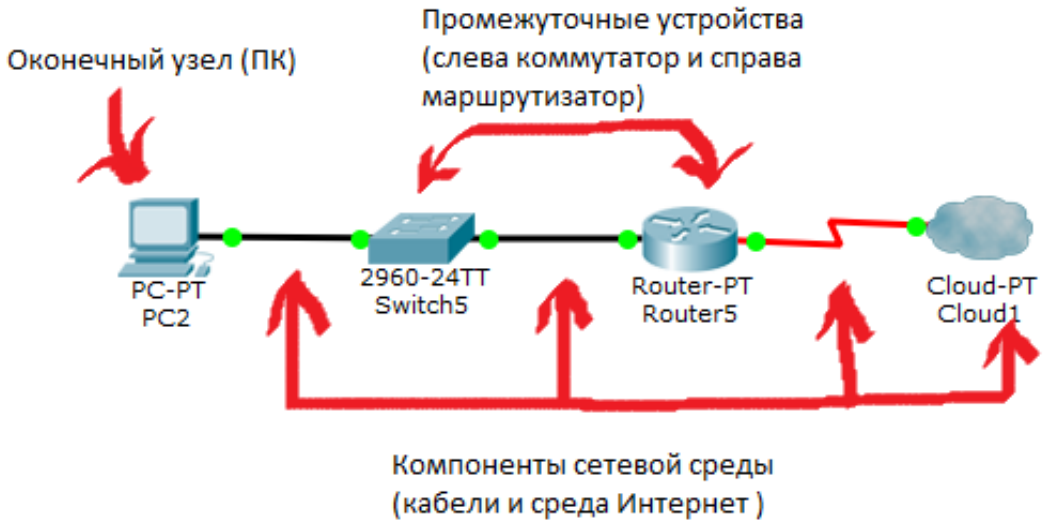
Что такое сеть? Это совокупность устройств и систем, которые подключены друг к другу (логически или физически) и общающихся между собой. Сюда можно отнести сервера, компьютеры, телефоны, маршрутизаторы и так далее. Размер этой сети может достигать размера Интернета, а может состоять всего из двух устройств, соединенных между собой кабелем. Чтобы не было каши, разделим компоненты сети на группы:

1) Оконечные узлы: Устройства, которые передают и/или принимают какие-либо данные. Это могут быть компьютеры, телефоны, сервера, какие-то терминалы или тонкие клиенты, телевизоры.

2) Промежуточные устройства: Это устройства, которые соединяют оконечные узлы между собой. Сюда можно отнести коммутаторы, концентраторы, модемы, маршрутизаторы, точки доступа Wi-Fi.

3) Сетевые среды: Это те среды, в которых происходит непосредственная передача данных. Сюда относятся кабели, сетевые карточки, различного рода коннекторы, воздушная среда передачи. Если это медный кабель, то передача данных осуществляется при помощи электрических сигналов. У оптоволоконных кабелей, при помощи световых импульсов. Ну и у беспроводных устройств, при помощи радиоволн.

Посмотрим все это на картинке:



На данный момент надо просто понимать отличие. Детальные отличия будут разобраны позже.

Теперь, на мой взгляд, главный вопрос: Для чего мы используем сети? Ответов на этот вопрос много, но я освещу самые популярные, которые используются в повседневной жизни:

1) Приложения: При помощи приложений отправляем разные данные между устройствами, открываем доступ к общим ресурсам. Это могут быть как консольные приложения, так и приложения с графическим интерфейсом.

2) Сетевые ресурсы: Это сетевые принтеры, которыми, к примеру, пользуются в офисе или сетевые камеры, которые просматривает охрана, находясь в удаленной местности.

3) Хранилище: Используя сервер или рабочую станцию, подключенную к сети, создается хранилище доступное для других. Многие люди выкладывают туда свои файлы, видео, картинки и открывают общий доступ к ним для других пользователей. Пример, который на ходу приходит в голову, — это google диск, яндекс диск и тому подобные сервисы.

4) Резервное копирование: Часто, в крупных компаниях, используют центральный сервер, куда все компьютеры копируют важные файлы для резервной копии. Это нужно для последующего восстановления данных, если оригинал удалился или повредился. Методов копирования огромное количество: с предварительным сжатием, кодированием и так далее.

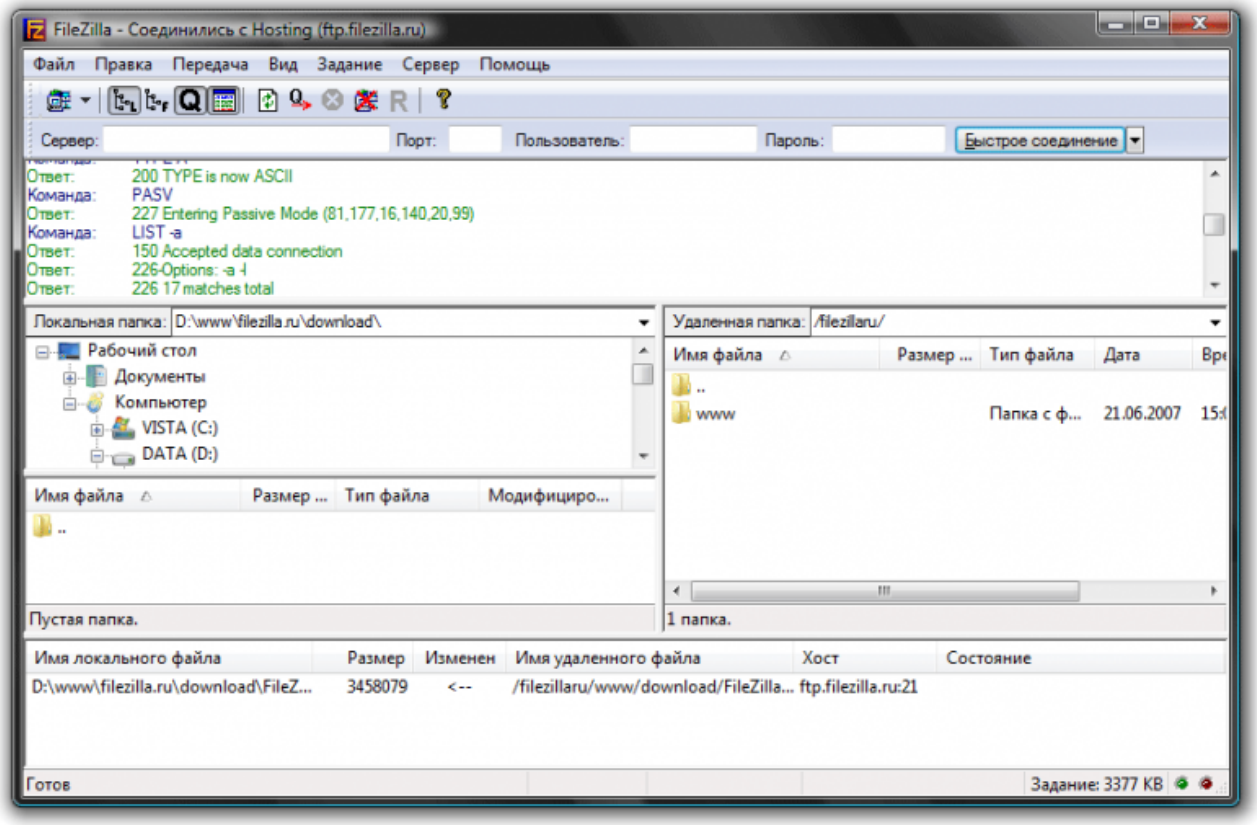
5) VoIP: Телефония, работающая по протоколу IP. Применяется она сейчас повсеместно, так как проще, дешевле традиционной телефонии и с каждым годом вытесняет ее.

Из всего списка, чаще всего многие работали именно с приложениями. Поэтому разберем их более подробно. Я старательно буду выбирать только те приложения, которые как-то связаны с сетью. Поэтому приложения типа калькулятора или блокнота, во внимание не беру.

1) Загрузчики. Это файловые менеджеры, работающие по протоколу FTP, TFTP. Банальный пример — это скачивание фильма, музыки, картинок с файлообменников или иных источников. К этой категории еще можно отнести резервное копирование, которое автоматически делает сервер каждую ночь. То есть это встроенные или сторонние программы и утилиты, которые выполняют копирование и скачивание. Данный вид приложений не требует прямого человеческого вмешательства. Достаточно указать место, куда сохранить и скачивание само начнется и закончится.

Скорость скачивания зависит от пропускной способности. Для данного типа приложений это не совсем критично. Если, например, файл будет скачиваться не минуту, а 10, то тут только вопрос времени, и на целостности файла это никак не скажется. Сложности могут возникнуть только когда нам надо за пару часов сделать резервную копию системы, а из-за плохого канала и, соответственно, низкой пропускной способности, это занимает несколько дней. Ниже приведены описания самых популярных протоколов данной группы:

FTP- это стандартный протокол передачи данных с установлением соединения. Работает по протоколу TCP (этот протокол в дальнейшем будет подробно рассмотрен). Стандартный номер порта 21. Чаще всего используется для загрузки сайта на веб-хостинг и выгрузки его. Самым популярным приложением, работающим по этому протоколу — это Filezilla. Вот так выглядит само приложение:



TFTP- это упрощенная версия протокола FTP, которая работает без установления соединения, по протоколу UDP. Применяется для загрузки образа бездисковыми рабочими станциями. Особенно широко используется устройствами Cisco для той же загрузки образа и резервных копий.

Интерактивные приложения. Приложения, позволяющие осуществить интерактивный обмен. Например, модель «человек-человек». Когда два человека, при помощи интерактивных приложений, общаются между собой или ведут общую работу. Сюда относится: ICQ, электронная почта, форум, на котором несколько экспертов помогают людям в решении вопросов. Или модель «человек-машина». Когда человек общается непосредственно с компьютером. Это может быть удаленная настройка базы, конфигурация сетевого устройства. Здесь, в отличие от загрузчиков, важно постоянное вмешательство человека. То есть, как минимум, один человек выступает инициатором. Пропускная способность уже более чувствительна к задержкам, чем приложения-загрузчики. Например, при удаленной конфигурации сетевого устройства, будет тяжело его настраивать, если отклик от команды будет в 30 секунд.

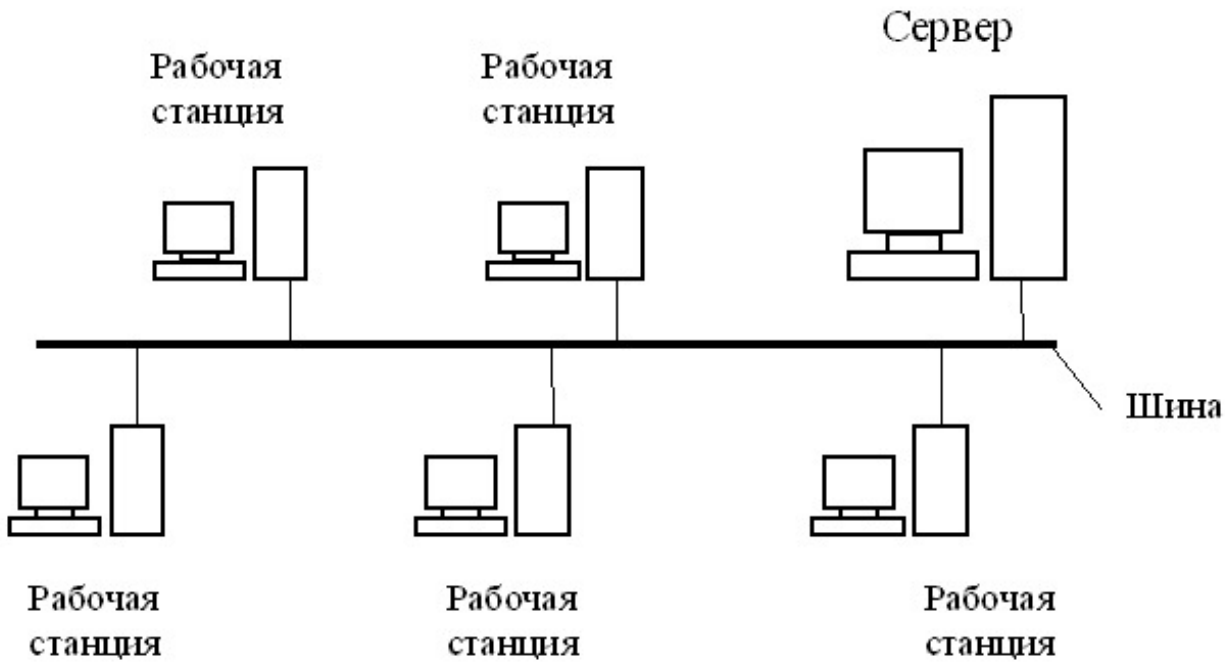
Приложения в реальном времени. Приложения, позволяющие передавать информацию в реальном времени. Как раз к этой группе относится IP-телефония, системы потокового вещания, видеоконференции. Самые чувствительные к задержкам и пропускной способности приложения. Представьте, что вы разговариваете по телефону и то, что вы говорите, собеседник услышит через 2 секунды и наоборот, вы от собеседника с таким же интервалом. Такое общение еще и приведет к тому, что голоса будут пропадать и разговор будет трудноразличимым,

а в видеоконференция превратится в кашу. В среднем, задержка не должна превышать 300 мс. К данной категории можно отнести Skype, Lync, Viber (когда совершаем звонок).

Теперь поговорим о такой важной вещи, как топология. Она делится на 2 большие категории: **физическая** и **логическая**. Очень важно понимать их разницу. Итак, **физическая** топология — это как наша сеть выглядит. Где находятся узлы, какие сетевые промежуточные устройства используются и где они стоят, какие сетевые кабели используются, как они протянуты и в какой порт воткнуты. **Логическая** топология — это каким путем будут идти пакеты в нашей физической топологии. То есть физическая — это как мы расположили устройства, а логическая — это через какие устройства будут проходить пакеты.

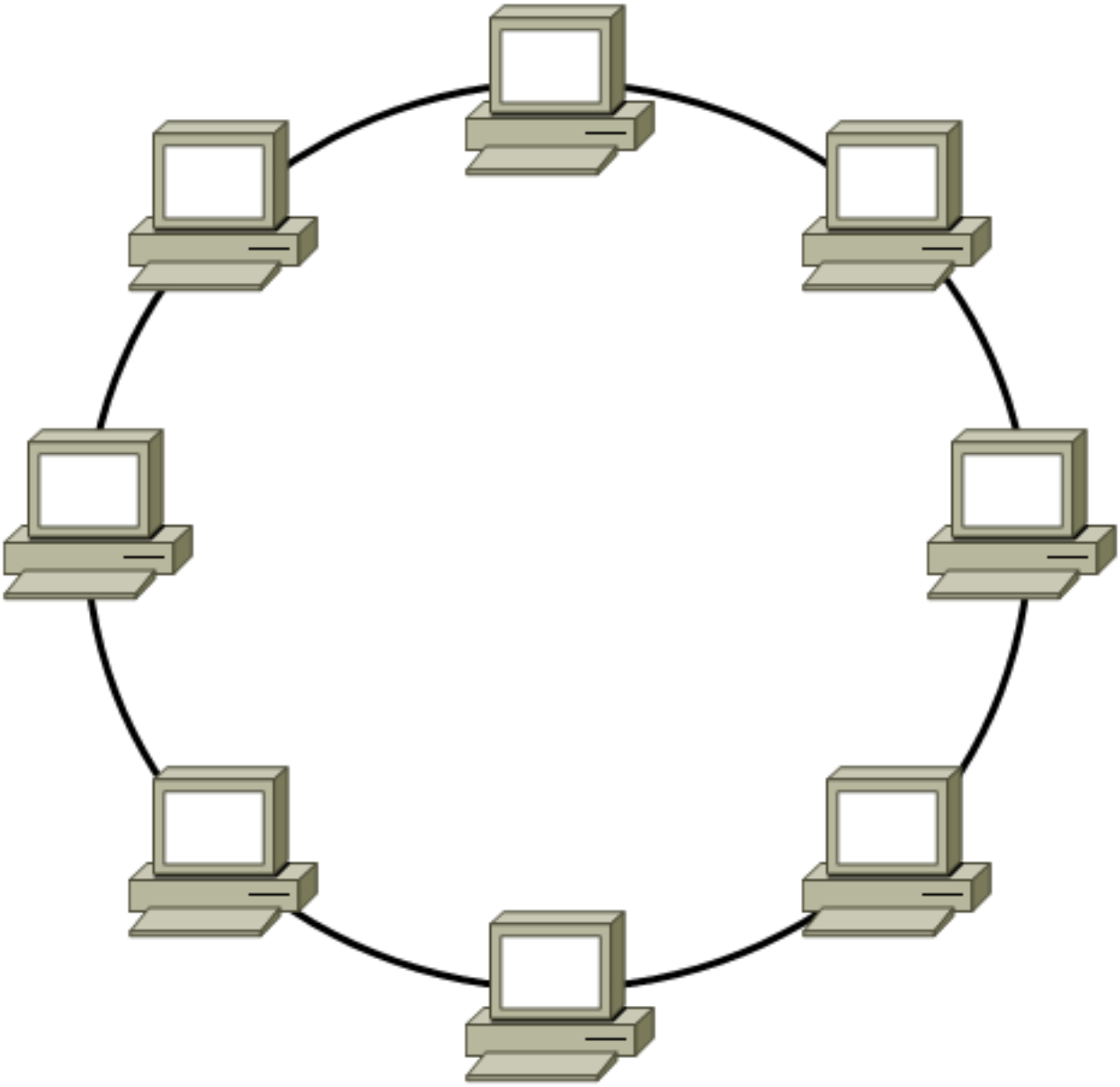
Теперь посмотрим и разберем виды топологии:

1) Топология с общей шиной (англ. Bus Topology)



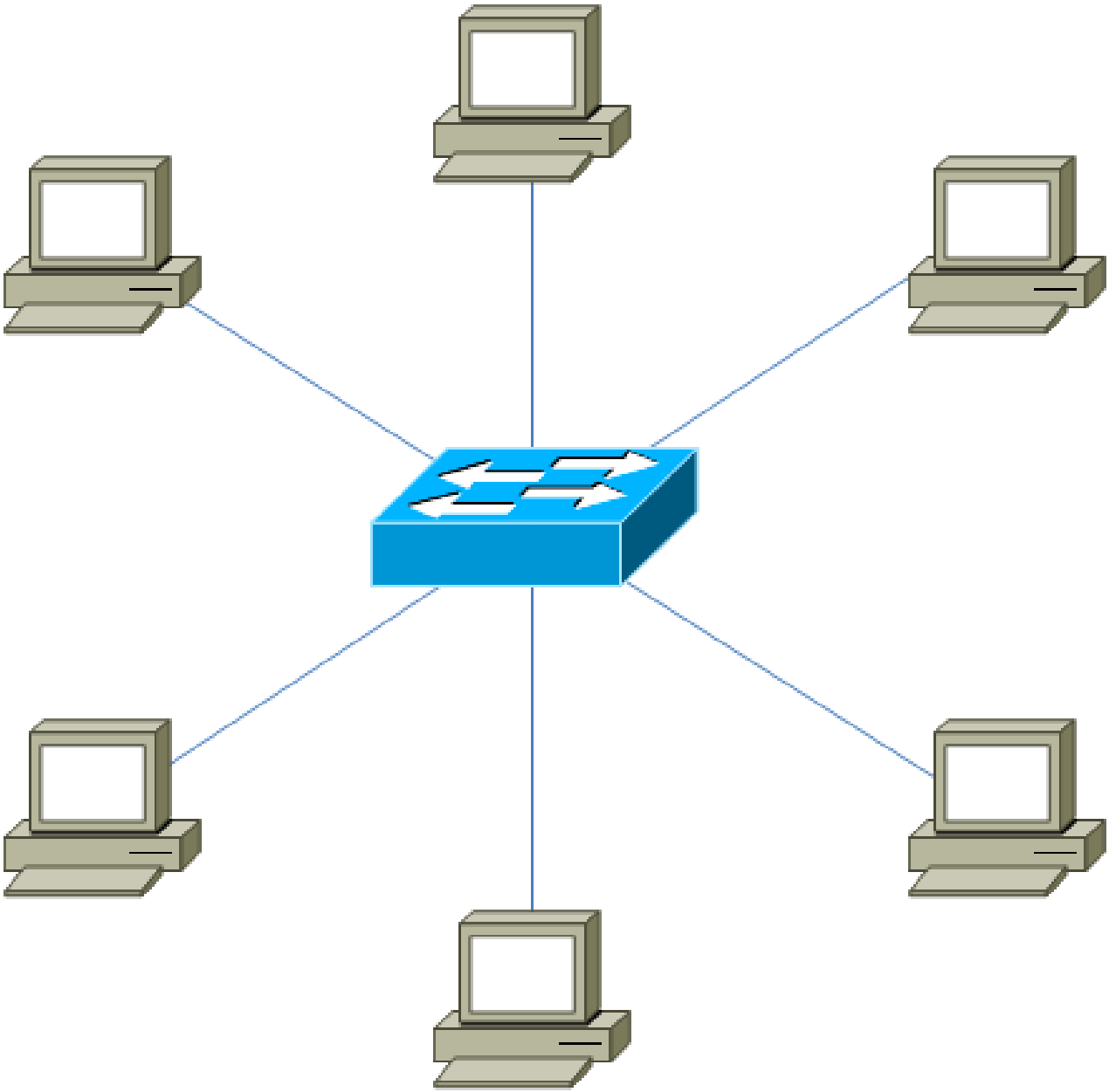
Одна из первых физических топологий. Суть состояла в том, что к одному длинному кабелю подсоединяли все устройства и организовывали локальную сеть. На концах кабеля требовались терминаторы. Как правило — это было сопротивление на 50 Ом, которое использовалось для того, чтобы сигнал не отражался в кабеле. Преимущество ее было только в простоте установки. С точки зрения работоспособности была крайне не устойчивой. Если где-то в кабеле происходил разрыв, то вся сеть оставалась парализованной, до замены кабеля.

2) Кольцевая топология (англ. Ring Topology)



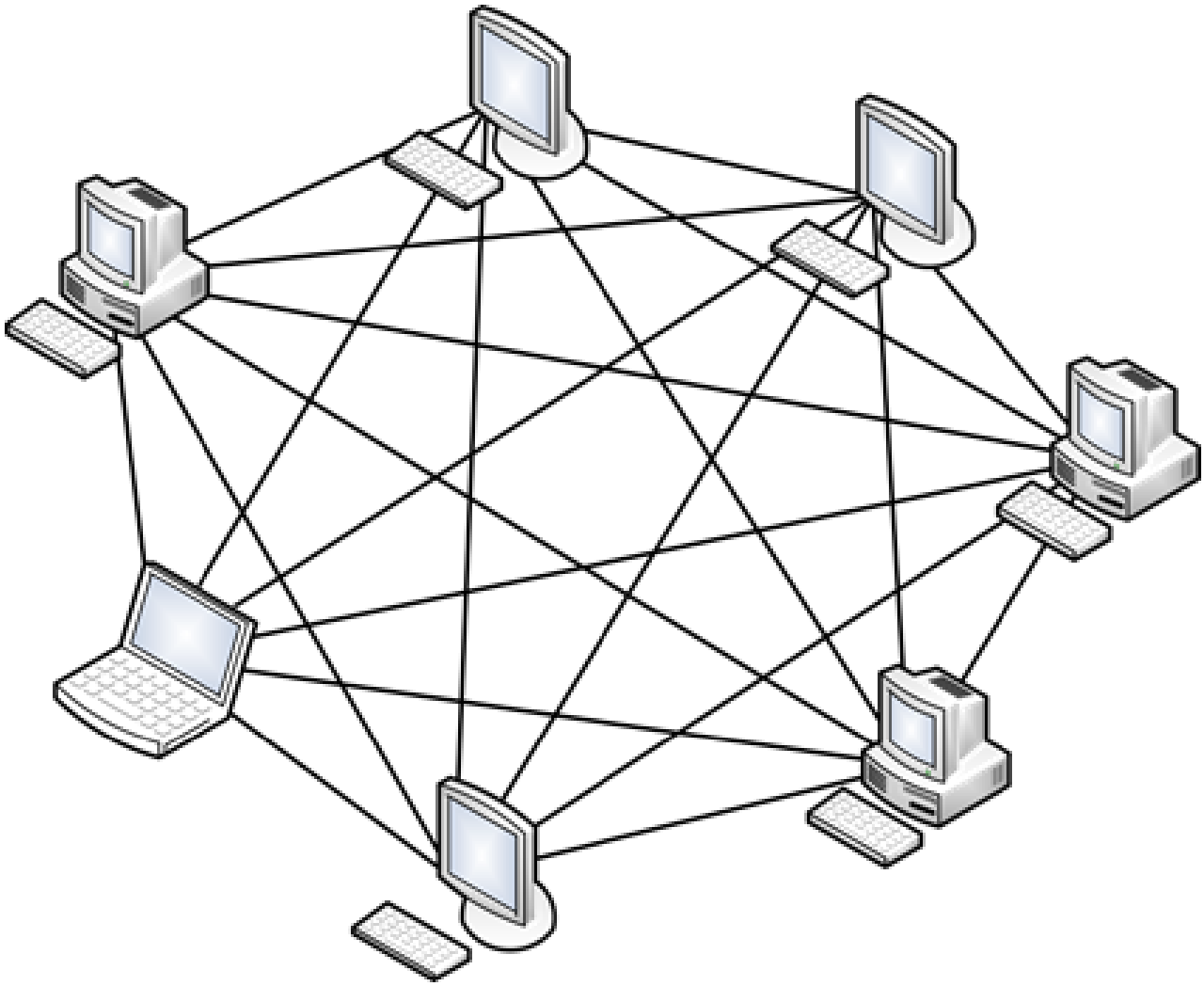
В данной топологии каждое устройство подключается к 2-ум соседним. Создавая, таким образом, кольцо. Здесь логика такова, что с одного конца компьютер только принимает, а с другого только отправляет. То есть, получается передача по кольцу и следующий компьютер играет роль ретранслятора сигнала. За счет этого нужда в терминаторах отпала. Соответственно, если где-то кабель повреждался, кольцо размыкалось и сеть становилась не работоспособной. Для повышения отказоустойчивости, применяют двойное кольцо, то есть в каждое устройство приходит два кабеля, а не один. Соответственно, при отказе одного кабеля, остается работать резервный.

3) Топология звезда (англ. Star Topology)



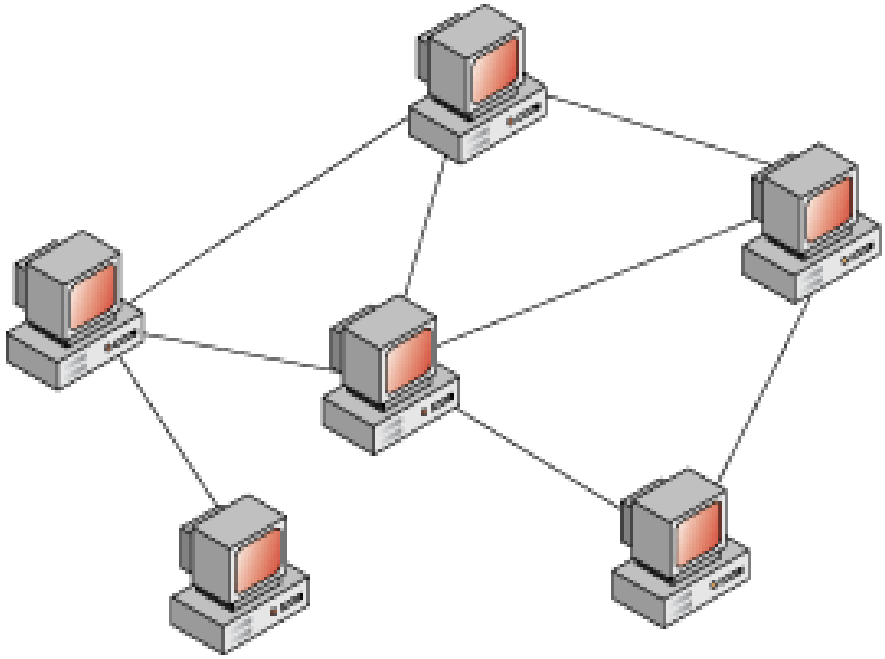
Все устройства подключаются к центральному узлу, который уже является ретранслятором. В наше время данная модель используется в локальных сетях, когда к одному коммутатору подключаются несколько устройств, и он является посредником в передаче. Здесь отказоустойчивость значительно выше, чем в предыдущих двух. При обрыве, какого либо кабеля, выпадает из сети только одно устройство. Все остальные продолжают спокойно работать. Однако если откажет центральное звено, сеть станет неработоспособной.

4)Полносвязная топология (англ. Full-Mesh Topology)



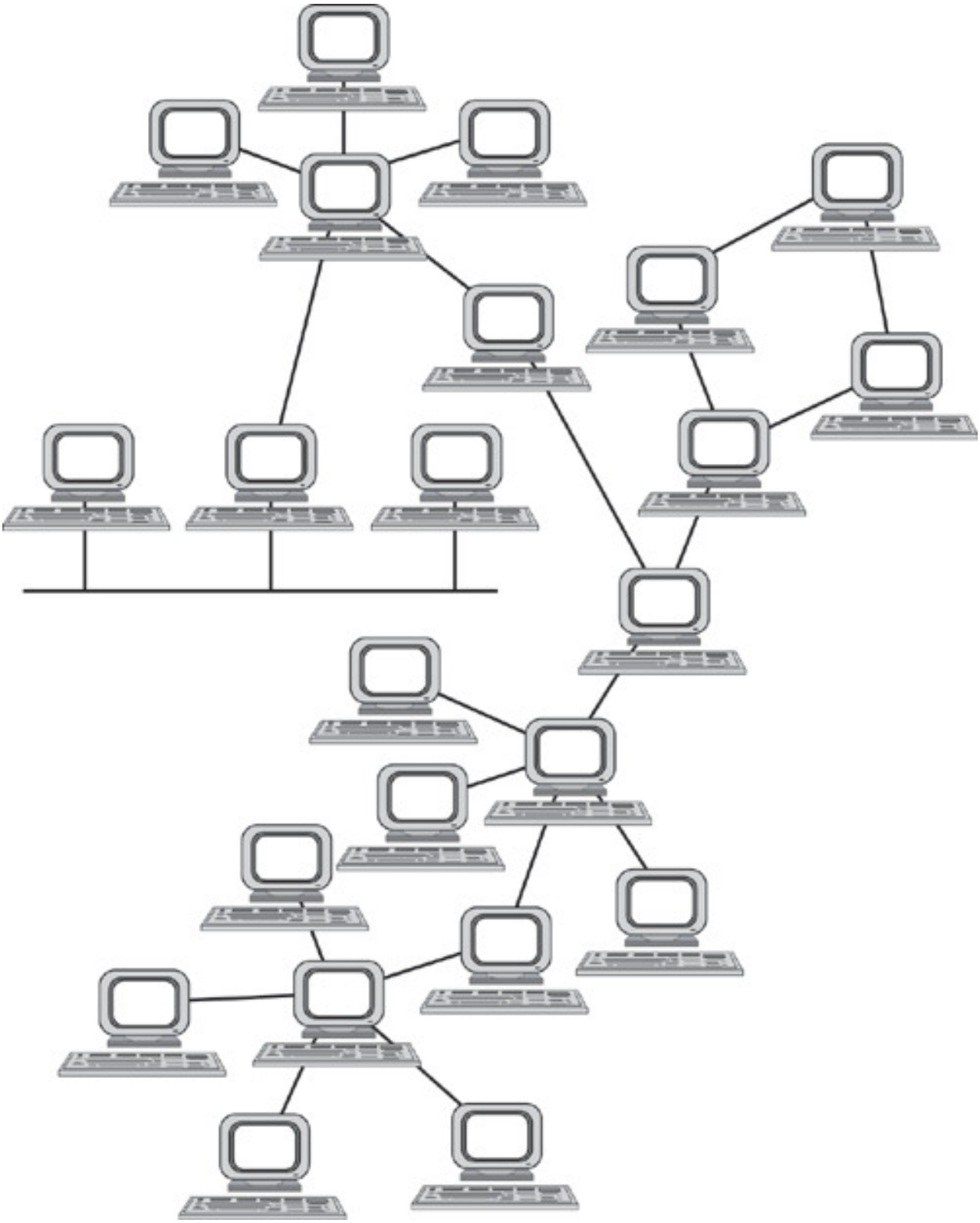
Все устройства связаны напрямую друг с другом. То есть с каждого на каждый. Данная модель является, пожалуй, самой отказоустойчивой, так как не зависит от других. Но строить сети на такой модели сложно и дорого. Так как в сети, в которой минимум 1000 компьютеров, придется подключать 1000 кабелей на каждый компьютер.

5)Неполносвязная топология (англ. Partial-Mesh Topology)



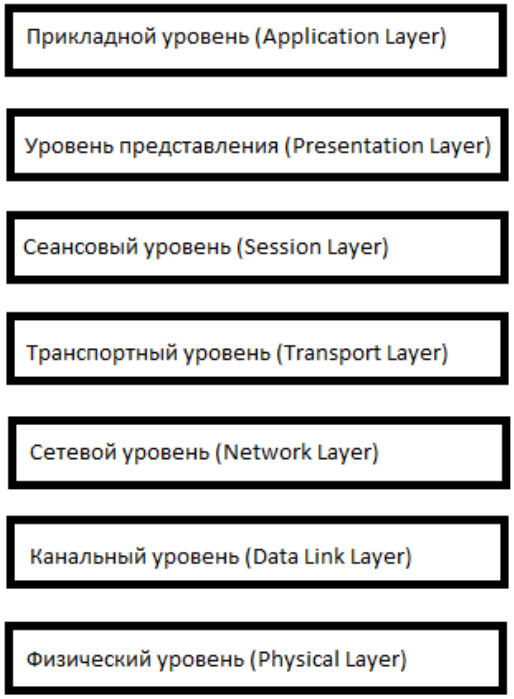
Как правило, вариантов ее несколько. Она похожа по строению на полносвязную топологию. Однако соединение построено не с каждого на каждый, а через дополнительные узлы. То есть узел А, связан напрямую только с узлом В, а узел В связан и с узлом А, и с узлом С. Так вот, чтобы узлу А отправить сообщение узлу С, ему надо отправить сначала узлу В, а узел В в свою очередь отправит это сообщение узлу С. В принципе по этой топологии работают маршрутизаторы. Приведу пример из домашней сети. Когда вы из дома выходите в Интернет, у вас нет прямого кабеля до всех узлов, и вы отправляете данные своему провайдеру, а он уже знает куда эти данные нужно отправить.

6) Смешанная топология (англ. Hybrid Topology)



Самая популярная топология, которая объединила все топологии выше в себя. Представляет собой древовидную структуру, которая объединяет все топологии. Одна из самых отказоустойчивых топологий, так как если у двух площадок произойдет обрыв, то парализована будет связь только между ними, а все остальные объединенные площадки будут работать безотказно. На сегодняшний день, данная топология используется во всех средних и крупных компаниях.

И последнее, что осталось разобрать — это сетевые модели. На этапе зарождения компьютеров, у сетей не было единых стандартов. Каждый вендор использовал свои проприетарные решения, которые не работали с технологиями других вендоров. Конечно, оставлять так было нельзя и нужно было придумывать общее решение. Эту задачу взвалила на себя международная организация по стандартизации (ISO — International Organization for Standartization). Они изучали многие, применяемые на то время, модели и в результате придумали **модель OSI**, релиз которой состоялся в 1984 году. Проблема ее была только в том, что ее разрабатывали около 7 лет. Пока специалисты спорили, как ее лучше сделать, другие модели модернизировались и набирали обороты. В настоящее время модель OSI не используют. Она применяется только в качестве обучения сетям. Мое личное мнение, что модель OSI должен знать каждый уважающий себя админ как таблицу умножения. Хоть ее и не применяют в том виде, в каком она есть, принципы работы у всех моделей схожи с ней.



Состоит она из 7 уровней и каждый уровень выполняет определенную ему роль и задачи. Разберем, что делает каждый уровень снизу вверх:

1) Физический уровень (Physical Layer): определяет метод передачи данных, какая среда используется (передача электрических сигналов, световых импульсов или радиоэфир), уровень напряжения, метод кодирования двоичных сигналов.

2) Канальный уровень (Data Link Layer): он берет на себя задачу адресации в пределах локальной сети, обнаруживает ошибки, проверяет целостность данных. Если слышали про MAC-адреса и протокол «Ethernet», то они располагаются на этом уровне.

3) Сетевой уровень (Network Layer): этот уровень берет на себя объединения участков сети и выбор оптимального пути (т.е. маршрутизация). Каждое сетевое устройство должно иметь уникальный сетевой адрес в сети. Думаю, многие слышали про протоколы IPv4 и IPv6. Эти протоколы работают на данном уровне.

4) Транспортный уровень (Transport Layer): Этот уровень берет на себя функцию транспорта. К примеру, когда вы скачиваете файл с Интернета, файл в виде сегментов отправляется на Ваш компьютер. Также здесь вводятся понятия портов, которые нужны для указания назначения к конкретной службе. На этом уровне работают протоколы TCP (с установлением соединения) и UDP (без установления соединения).

5) Сеансовый уровень (Session Layer): Роль этого уровня в установлении, управлении и разрыве соединения между двумя хостами. К примеру, когда открываете страницу на веб-сервере, то Вы не единственный посетитель на нем. И вот для того, чтобы поддерживать сеансы со всеми пользователями, нужен сеансовый уровень.

6) Уровень представления (Presentation Layer): Он структурирует информацию в читабельный вид для прикладного уровня. Например, многие компьютеры используют таблицу кодировки ASCII для вывода текстовой информации или формат jpeg для вывода графического изображения.

7) Прикладной уровень (Application Layer): Наверное, это самый понятный для всех уровень. Как раз на этом уроне работают привычные для нас приложения — e-mail, браузеры по протоколу HTTP, FTP и остальное.

Самое главное помнить, что нельзя перескакивать с уровня на уровень (Например, с прикладного на канальный, или с физического на транспортный). Весь путь должен проходить строго с верхнего на нижний и с нижнего на верхний. Такие процессы получили название **инкапсуляция** (с верхнего на нижний) и **деинкапсуляция** (с нижнего на верхний). Также стоит упомянуть, что на каждом уровне передаваемая информация называется по-разному.

На прикладном, представления и сеансовым уровнях, передаваемая информация обозначается как PDU (Protocol Data Units). На русском еще называют блоки данных, хотя в моем круге их называют просто данные).

Информацию транспортного уровня называют сегментами. Хотя понятие сегменты, применимо только для протокола TCP. Для протокола UDP используется понятие — датаграмма. Но, как правило, на это различие закрывают глаза.
На сетевом уровне называют IP пакеты или просто пакеты.

И на канальном уровне — кадры. С одной стороны это все терминология и она не играет важной роли в том, как вы будете называть передаваемые данные, но для экзамена эти понятия лучше знать. Итак, приведу свой любимый пример, который помог мне, в мое время, разобраться с процессом инкапсуляции и деинкапусуляции:

1) Представим ситуацию, что вы сидите у себя дома за компьютером, а в соседней комнате у вас свой локальный веб-сервер. И вот вам понадобилось скачать файл с него. Вы набираете адрес страницы вашего сайта. Сейчас вы используете протокол HTTP, которые работает на прикладном уровне. Данные упаковываются и спускаются на уровень ниже.

2) Полученные данные прибегают на уровень представления. Здесь эти данные структурируются и приводятся в формат, который сможет быть прочитан на сервере. Запаковывается и спускается ниже.

3) На этом уровне создается сессия между компьютером и сервером.

4) Так как это веб сервер и требуется надежное установление соединения и контроль за принятыми данными, используется протокол TCP. Здесь мы указываем порт, на который будем стучаться и порт источника, чтобы сервер знал, куда отправлять ответ. Это нужно для того, чтобы сервер понял, что мы хотим попасть на веб-сервер (стандартно — это 80 порт), а не на почтовый сервер. Упаковываем и спускаем дальше.

5) Здесь мы должны указать, на какой адрес отправлять пакет. Соответственно, указываем адрес назначения (пусть адрес сервера будет 192.168.1.2) и адрес источника (адрес компьютера 192.168.1.1). Заворачиваем и спускаем дальше.

6) IP пакет спускается вниз и тут вступает в работу канальный уровень. Он добавляет физические адреса источника и назначения, о которых подробно будет расписано в последующей статье. Так как у нас компьютер и сервер в локальной среде, то адресом источника будет являться MAC-адрес компьютера, а адресом назначения MAC-адрес сервера (если бы компьютер и сервер находились в разных сетях, то адресация работала по-другому). Если на верхних уровнях каждый раз добавлялся заголовок, то здесь еще добавляется концевик, который указывает на конец кадра и готовность всех собранных данных к отправке.

7) И уже физический уровень конвертирует полученное в биты и при помощи электрических сигналов (если это витая пара), отправляет на сервер.

Процесс деинкапсуляции аналогичен, но с обратной последовательностью:

1) На физическом уровне принимаются электрические сигналы и конвертируются в понятную битовую последовательность для канального уровня.

2) На канальном уровне проверяется MAC-адрес назначения (ему ли это адресовано). Если да, то проверяется кадр на целостность и отсутствие ошибок, если все прекрасно и данные целы, он передает их вышестоящему уровню.

3) На сетевом уровне проверяется IP адрес назначения. И если он верен, данные поднимаются выше. Не стоит сейчас вдаваться в подробности, почему у нас адресация на канальном и сетевом уровне. Это тема требует особого внимания, и я подробно объясню их различие позже. Главное сейчас понять, как данные упаковываются и распаковываются.

4) На транспортном уровне проверяется порт назначения (не адрес). И по номеру порта, выясняется какому приложению или сервису адресованы данные. У нас это веб-сервер и номер порта — 80.

5) На этом уровне происходит установление сеанса между компьютером и сервером.

6) Уровень представления видит, как все должно быть структурировано и приводит информацию в читабельный вид.

7) И на этом уровне приложения или сервисы понимают, что надо выполнить.

Много было написано про модель OSI. Хотя я постарался быть максимально краток и осветить самое важное. На самом деле про эту модель в Интернете и в книгах написано очень много и подробно, но для новичков и готовящихся к CCNA, этого достаточно. Из вопросов на экзамене по данной модели может быть 2 вопроса. Это правильно расположить уровни и на каком уровне работает определенный протокол.

Как было написано выше, модель OSI в наше время не используется. Пока разрабатывалась эта модель, все большую популярность получал стек протоколов TCP/IP. Он был значительно проще и завоевал быструю популярность.
Вот так этот стек выглядит:

Как видно, он отличается от OSI и даже сменил название некоторых уровней. По сути, принцип у него тот же, что и у OSI. Но только три верхних уровня OSI: прикладной, представления и сеансовый объединены у TCP/IP в один, под названием прикладной. Сетевой уровень сменил название и называется — Интернет. Транспортный остался таким же и с тем же названием. А два нижних уровня OSI: канальный и физический объединены у TCP/IP в один с названием — уровень сетевого доступа. Стек TCP/IP в некоторых источниках обозначают еще как модель DoD (Department of Defence). Как говорит википедия, была разработана Министерством обороны США. Этот вопрос встретился мне на экзамене и до этого я про нее ничего не слышал. Соответственно вопрос: «Как называется сетевой уровень в модели DoD?», ввел меня в ступор. Поэтому знать это полезно.

Было еще несколько сетевых моделей, которые, какое то время держались. Это был стек протоколов IPX/SPX. Использовался с середины 80-х годов и продержался до конца 90-х, где его вытеснила TCP/IP. Был реализован компанией Novell и являлся модернизированной версией стека протоколов Xerox Network Services компании Xerox. Использовался в локальных сетях долгое время. Впервые IPX/SPX я увидел в игре «Казаки». При выборе сетевой игры, там предлагалось несколько стеков на выбор. И хоть выпуск этой игры был, где то в 2001 году, это говорило о том, что IPX/SPX еще встречался в локальных сетях.

Еще один стек, который стоит упомянуть — это AppleTalk. Как ясно из названия, был придуман компанией Apple. Создан был в том же году, в котором состоялся релиз модели OSI, то есть в 1984 году. Продержался он совсем недолго и Apple решила использовать вместо него TCP/IP.

Также хочу подчеркнуть одну важную вещь. Token Ring и FDDI — не сетевые модели! Token Ring — это протокол канального уровня, а FDDI это стандарт передачи данных, который как раз основывается на протоколе Token Ring. Это не самая важная информация, так как эти понятия сейчас не встретишь. Но главное помнить о том, что это не сетевые модели.

Вот и подошла к концу статья по первой теме. Хоть и поверхностно, но было рассмотрено много понятий. Самые ключевые будут разобраны подробнее в следующих статьях. Надеюсь теперь сети перестанут казаться чем то невозможным и страшным, а читать умные книги будет легче). Если я что-то забыл упомянуть, возникли дополнительные вопросы или у кого есть, что дополнить к этой статье, оставляйте комментарии, либо спрашивайте лично. Спасибо за прочтение. Буду готовить следующую тему.