**Топология локальных вычислительных сетей**

**Топология ЛВС типа звезда**  
  
Концепция данной топологии ЛВС была заимствована из области больших ЭВМ, когда все данные, полученные с периферийных устройств, обрабатываются головной машиной. При этом обмен данными между периферийными рабочими местами происходит через головную машину, которая является центральным узлом ЛВС. Такая топология локальной вычислительной сети используется, к примеру, электронной почтой RELCOM.  
  
Пропускная способность такой локальной вычислительной гарантирована для каждой рабочей станции сети и зависит только от вычислительной мощности узла. Возникновение коллизий в сети такой топологии невозможно.   
  
Сети, построенные по топологии «звезда» имеют максимально возможное быстродействие, так как данных между рабочими станциями передаются через центральный узел по отдельным линиям, которые используются исключительно этими станциями. Частота запросов для передачи информации между станциями относительно невелика.   
  
Производительность ЛВС находится в прямой зависимости от мощности файлового сервера. Если центральный узел выходит из строя, сеть также прекращает работу.   
  
Монтаж кабельного соединения несложен, поскольку каждая рабочая станция связана только с головной машиной, но общая стоимость кабеля может оказаться достаточно большой, и увеличивается в случае расположения главной машины не в центре сети.  
  
Для расширения сети необходим монтаж отдельного кабеля от новой рабочей станции к головной машине.   
  
Управление сетью осуществляется из ее центра, в центре же реализуется механизм защиты информации.   
  
**Кольцевая топология ЛВС**  
  
Кольцевой топологии локальной вычислительной сети подразумевает, что рабочие станции связаны друг с другом по кругу: первая со второй, третья с четвертой и так далее. Последняя станция связывается с первой, замыкая кольцо.  
  
Сложность и стоимость монтажа кабелей между рабочими станциями напрямую зависит от географического расположения станций друг относительно друга.   
  
Передача информации осуществляется по кругу. Рабочая станция получает запрос из кольца, а затем отправляет информацию по конкретному адресу. Система передачи информации такой ЛВС считается достаточно эффективной, поскольку сообщения можно отправлять друг за другом достаточно быстро, кроме того легко отправить запрос на все рабочие станции кольца. Время передачи информации растет с увеличением количества станций в ЛВС.   
  
Недостаток локальных сетей с кольцевой топологией в том, что при выходе из строя хотя бы одной рабочей станции вся сеть становится неработоспособной. Любую неисправность кабельного соединения в такой сети обнаружить несложно.   
  
Для подключения новой станции в локальную сеть необходимо временное отключение сети. Протяженность такой сети может быть неограниченной.   
  
**Логическая кольцевая локальная вычислительная сеть**  
  
Логическая кольцевая локальная вычислительная сеть является специальной формой топологии ЛВС. Она представляет собой соединение нескольких сетей, организованных по топологии звезда. Для подключения в сеть отдельных «звезд» используются специальные концентраторы, которые часто называют хабами. Хабы могут быть активными либо пассивными. Отличие активных концентраторов – в наличии дополнительного усилителя, которых служит для подключения 4 - 16 рабочих станций. Пассивный концентратор рассчитан на три рабочих станции и по своей сути является просто разветвительным устройством. Управление каждой конкретной станцией в сети осуществляется точно так же, как в кольцевой ЛВС. Каждая рабочая станция сети получает собственный адрес, по которому и осуществляется передача управления. Сбой в работе одной из машин может повлиять только на нижестоящие станции, выход из строя всей сети маловероятен.   
  
**Шинная топология ЛВС**  
  
Шинная топология сети предполагает, что средой для передачи данных служит коммуникационный путь, к которому подключены все рабочие станции. Каждая из станций сети может вступить в непосредственный контакт с любой другой станцией ЛВС.   
  
Подключение или отключение рабочих станций осуществляется без прерывания работы ЛВС, состояние отдельных рабочих станций на работоспособность сети в целом не влияет.

|  |  |
| --- | --- |
| |  | | --- | | **Понятие сетевая архитектура и Топология сети** | |
| |  | | --- | | **Сетевая архитектура** (network architecture) – это комбинация топологий, методов доступа к среде передачи данных и протоколов, необходимых для создания работоспособной сети.   Термин **сетевая топология** обозначает физическое расположение компьютеров, кабелей и других сетевых компонентов. Существуют три базовые топологии сети:  -Шина (Bus)  -Звезда (Star)  - Кольцо (Ring) Шина (Bus) http://www.servicecall.ru/Images_network/Picture10.png  Используется один кабель, именуемый магистралью или сегментом, к которому подключены все компьютеры сети. Данные передаются всем компьютерам сети, однако информацию принимает только один компьютер, чей адрес соответствует адресу получателя, присутствующему среди передаваемых данных. В каждый момент времени передачу может вести только один компьютер.  **Шина** – пассивная топология. Компьютеры не перемещают данные от отправителя к получателю. Если один компьютер выходит из строя, это не скажется на работе сети. В активных топологиях компьютеры регенерируют сигналы и передают их дальше по сети. Звезда (Star) http://www.servicecall.ru/Images_network/Picture11.png  Все компьютеры с помощью сегментов кабеля подключаются к центральному устройству. При выходе из строя одного компьютера или одного сегмента кабеля, только этот компьютер не работает в сети. Если центральный компонент выходит из строя, не работает вся сеть. Кольцо (Ring) http://www.servicecall.ru/Images_network/Picture13.png Логическое кольцо - Физическая звезда http://www.servicecall.ru/Images_network/Picture14.png Ячеистая топология (mesh topology)  Все компьютеры связаны друг с другом отдельными соединениями. Для локальных сетей существует скорее в виде теоретической концепции (реальный пример – соединение двух компьютеров).  http://www.servicecall.ru/Images_network/Picture15.png  В ГВС (интерсети) ячеистая топология используется. В такой сети благодаря использованию избыточных маршрутизаторов данные могут доставляться от одной системы к другой несколькими путями.  http://www.servicecall.ru/Images_network/Picture16.png | |

## Архитектура "клиент-сервер".

### Основные понятия.

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находится как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Основной принцип технологии "клиент-сервер" заключается в разделении функций приложения на три группы:

* ввод и отображение данных (взаимодействие с пользователем);
* прикладные функции, характерные для данной предметной области;
* функции управления ресурсами (файловой системой, базой даных и т.д.)

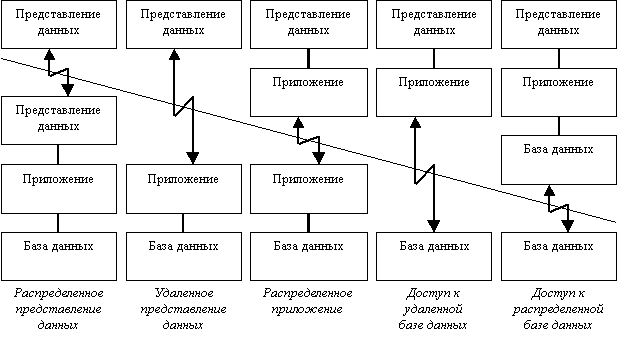
Поэтому, в любом приложении выделяются следующие компоненты:

* компонент представления данных
* прикладной компонент
* компонент управления ресурсом

Связь между компонентами осуществляется по определенным правилам, которые называют "протокол взаимодействия".

### Модели взаимодействия клиент-сервер.

Компанией [Gartner Group](http://www.gartner.com/), специализирующейся в области исследования информационных технологий, предложена следующая классификация двухзвенных моделей взаимодействия клиент-сервер (двухзвенными эти модели называются потому, что три компонента приложения различным образом распределяются между двумя узлами):



Исторически первой появилась модель распределенного представления данных, которая реализовывалась на универсальной ЭВМ с подключенными к ней неинтеллектуальными терминалами. Управление данными и взаимодействие с пользователем при этом объединялись в одной программе, на терминал передавалась только "картинка", сформированная на центральном компьютере.

Затем, с появлением персональных компьютеров (ПК) и локальных сетей, были реализованы модели доступа к удаленной базе данных. Некоторое время базовой для сетей ПК была архитектура файлового сервера. При этом один из компьютеров является файловым сервером, на клиентах выполняются приложения, в которых совмещены компонент представления и прикладной компонент (СУБД и прикладная программма). Протокол обмена при этом представляет набор низкоуровненых вызовов операций файловой системы. Такая архитектура, реализуемая, как правило, с помощью персональных СУБД (см. параграф [4.8](http://www.mstu.edu.ru/study/materials/zelenkov/ch_4_8.html)), имеет очевидные недостатки - высокий сетевой трафик и отсутствие унифицированного доступа к ресурсам.

С появлением первых специализированных серверов баз данных появилась возможность другой реализации модели доступа к удаленной базе данных. В этом случае ядро СУБД функционирует на сервере, протокол обмена обеспечивается с помощью языка SQL. Такой подход по сравнению с файловым сервером ведет к уменьшению загрузки сети и унификации интерфейса "клиент-сервер". Однако, сетевой трафик остается достаточно высоким, кроме того, по прежнему невозможно удовлетворительное администрирование приложений, поскольку в одной программе совмещаются различные функции.

Позже была разработана концепция активного сервера, который использовал механизм хранимых процедур. Это позволило часть прикладного компонента перенести на сервер (модель распределенного приложения). Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, что и SQL-сервер. Преимущества такого подхода: возможно централизованное администрирование прикладных функций, значительно снижается сетевой трафик (т.к. передаются не SQL-запросы, а вызовы хранимых процедур). Недостаток - ограниченность средств разработки хранимых процедур по сравнению с языками общего назначения (C и Pascal).

На практике сейчас обычно используются смешанный подход:

* простейшие прикладные функции выполняются хранимыми процедурами на сервере
* более сложные реализуются на клиенте непосредственно в прикладной программе

Сейчас ряд поставщиков коммерческих СУБД объявило о планах реализации механизмов выполнения хранимых процедур с использованием языка Java. Это соответствует концепции "тонкого клиента", функцией которого остается только отображение данных (модель удаленного представления данных).

В последнее время также наблюдается тенденция ко все большему использованию модели распределенного приложения. Характерной чертой таких приложений является логическое разделение приложения на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная архитектура клиент-сервер становится трехзвенной, а к некоторых случаях, она может включать и больше звеньев.

