13. Сравнительная характеристика протоколов с установкой и без установки соединения. Протокол UDP. Формат пакетов и принципы работы. Понятие порта.

Сравнительная характеристика протоколов:



В отличие от TCP UDP — очень быстрый протокол, поскольку в нем определен самый минимальный механизм, необходимый для передачи данных. Конечно, он имеет некоторые недостатки. Сообщения поступают в любом порядке, и то, которое отправлено первым, может быть получено последним. Доставка сообщений UDP вовсе не гарантируется, сообщение может потеряться, и могут быть получены две копии одного и того же сообщения. Последний случай возникает, если для отправки сообщений в один адрес использовать два разных маршрута.

UDP не требует открывать соединение, и данные могут быть отправлены сразу же, как только они подготовлены. UDP не отправляет подтверждающие сообщения, поэтому данные могут быть получены или потеряны. Если при использовании UDP требуется надежная передача данных, ее следует реализовать в протоколе более высокого уровня.

Структура UDP-пакета:



1. Номер порта отправителя 16 битов – задаёт порт, на который может

быть отправлен ответ, если таковой ожидается; иначе – 0.

1. Номер порта получателя 16 битов – обязательное поле.

3. Длина дейтаграммы 16 битов – размер заголовка и данных в байтах.

◼ Минимальная длина пакета равна длине заголовка – 8 байтов.

◼ Фактический максимум: 65 515 (-20 байтов на заголовок IPv4).

◼ Таким образом, фактический предел размера данных в UDP с

IPv4 – 65 507 (ещё -8 байтов заголовок UDP).

1. Контрольная сумма\* 16 битов – для предотвращения ошибочной

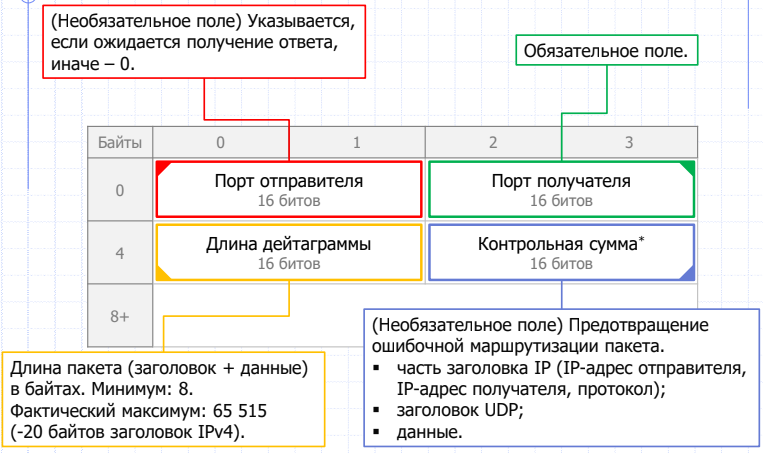
маршрутизации дейтаграмм; рассчитывается на основании части

заголовка IP (IP-адрес отправителя, IP-адрес получателя,

протокол), заголовка UDP и данных. Если контроль обеспечивается

протоколом вышестоящего уровня, заполняется нулями. Не

является обязательным для IPv4.



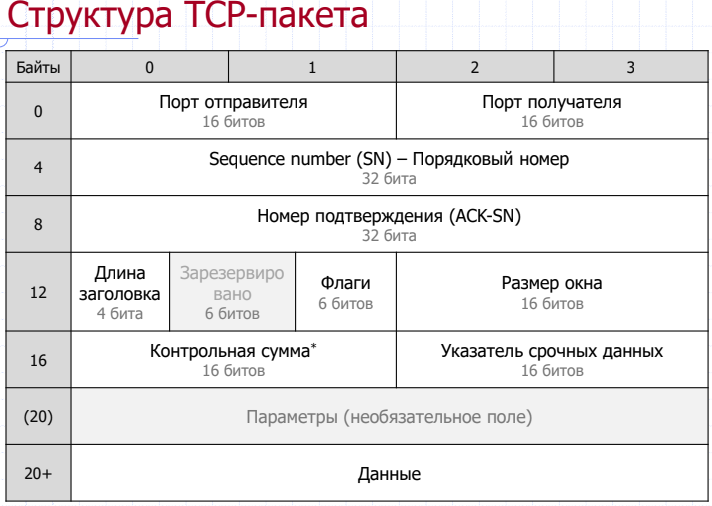
Порт - Логическая точка подключения для процесса. Процессы обмениваются данными через порты (или сокеты). У каждого порта есть очередь данных для отправки и очередь полученных данных. В сети, где применяются интерфейсы, с каждым портом, в зависимости от его назначения, связывается номер порта Internet. Отдельный порт идентифицируется по адресу сокета Internet, который представляет собой пару, состоящую из IP-адреса хоста и номера порта.

14. Протокол TCP. Формат пакетов и общие принципы работы.   
Обмен данными, ориентированный на соединения, может использовать надежную связь, для обеспечения которой протокол уровня 4 посылает подтверждения о получении данных и запрашивает повторную передачу, если данные не получены или искажены. Протокол TCP использует именно такую надежную связь. TCP используется в таких прикладных протоколах, как HTTP, FTP, SMTP и Telnet.

Протокол TCP требует, чтобы перед отправкой сообщения было открыто соединение. Серверное приложение должно выполнить так называемое пассивное открытие (passive open), чтобы создать соединение с известным номером порта, и, вместо того чтобы отправлять вызов в сеть, сервер переходит в ожидание поступления входящих запросов. Клиентское приложение должно выполнить активное открытие (active open), отправив серверному приложению синхронизирующий порядковый номер (SYN), идентифицирующий соединение. Клиентское приложение может использовать динамический номер порта в качестве локального порта.

Сервер должен отправить клиенту подтверждение (ACK) вместе с порядковым номером (SYN) сервера. В свою очередь клиент отвечает АСК, и соединение устанавливается.

После этого может начаться процесс отправки и получения сообщений. При получении сообщения в ответ всегда отправляется сообщение АСК. Если до получения АСК отправителем истекает тайм-аут, сообщение помещается в очередь на повторную передачу.



15. Принцип скользящего окна в протоколе TCP. Проблемы TCP.  
В рамках установленного соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя. Квитирование - это один из традиционных методов обеспечения надежной связи. В протоколе TCP используется частный случай квитирования - алгоритм скользящего окна. Идея этого алгоритма была изложена в главе 2, «Основы передачи дискретных данных».

Особенность использования алгоритма скользящего окна в протоколе TCP состоит в том, что, хотя единицей передаваемых данных является сегмент, окно определено на множестве нумерованных байтов неструктурированного потока данных, поступающих с верхнего уровня и буферизуемых протоколом TCP. Получающий модуль TCP отправляет «окно» посылающему модулю TCP. Данное окно задает количество байтов (начиная с номера байта, о котором уже была выслана квитанция), которое принимающий модуль TCP готов в настоящий момент принять.

Квитанция (подтверждение) посылается только в случае правильного приема данных, отрицательные квитанции не посылаются. Таким образом, отсутствие квитанции означает либо прием искаженного сегмента, либо потерю сегмента, либо потерю квитанции. В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в которое помещает число, на единицу превышающее максимальный номер байта в полученном сегменте. Это число часто называют номером очереди.

На рис. 5.24 показан поток байтов, поступающий на вход протокола TCP. Из потока байтов модуль TCP нарезает последовательность сегментов. Для определенности на рисунке принято направление перемещения данных справа налево. В этом потоке можно указать несколько логических границ. Первая граница отделяет сегменты, которые уже были отправлены и на которые уже пришли квитанции. Следующую часть потока составляют сегменты, которые также уже отправлены, так как входят в границы, определенные окном, но квитанции на них пока не получены. Третья часть потока - это сегменты, которые пока не отправлены, но могут быть отправлены, так как входят в пределы окна. И наконец, последняя граница указывает на начало последовательности сегментов, ни один из которых не может быть отправлен до тех пор, пока не придет очередная квитанция и окно не будет сдвинуто вправо.

Рис. 5.24. Особенности реализации алгоритма скользящего окна в протоколе TCP

Если размер окна равен W, а последняя по времени квитанция содержала значение N, то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером N+W. Этот сегмент выходит за рамки окна, и передачу в таком случае необходимо приостановить до прихода следующей квитанции.

Надежность передачи достигается благодаря подтверждениям и номерам очереди. Концептуально каждому байту данных присваивается номер очереди. Номер очереди для первого байта данных в сегменте передается вместе с этим сегментом и называется номером очереди для сегмента. Сегменты также несут номер подтверждения, который является номером для следующего ожидаемого байта данных, передаваемого в обратном направлении. Когда протокол TCP передает сегмент с данными, он помещает его копию в очередь повторной передачи и запускает таймер. Когда приходит подтверждение для этих данных, соответствующий сегмент удаляется из очереди. Если подтверждение не приходит до истечения срока, то сегмент посылается повторно.

Выбор времени ожидания (тайм-аута) очередной квитанции является важной задачей, результат решения которой влияет на производительность протокола TCP. Тайм-аут не должен быть слишком коротким, чтобы по возможности исключить избыточные повторные передачи, которые снижают полезную пропускную способность системы. Но он не должен быть и слишком большим, чтобы избежать длительных простоев, связанных с ожиданием несуществующей или «заблудившейся» квитанции.

При выборе величины тайм-аута должны учитываться скорость и надежность физических линий связи, их протяженность и многие другие подобные факторы. В протоколе TCP тайм-аут определяется с помощью достаточно сложного адаптивного алгоритма, идея которого состоит в следующем. При каждой передаче засекается время от момента отправки сегмента до прихода квитанции о его приеме (время оборота). Получаемые значения времени оборота усредняются с весовыми коэффициентами, возрастающими от предыдущего замера к последующему. Это делается с тем, чтобы усилить влияние последних замеров. В качестве тайм-аута выбирается среднее время оборота, умноженное на некоторый коэффициент. Практика показывает, что значение этого коэффициента должно превышать 2. В сетях с большим разбросом времени оборота при выборе тайм-аута учитывается и дисперсия этой величины.

Поскольку каждый байт пронумерован, то каждый из них может быть опознан. Приемлемый механизм опознавания является накопительным, поэтому опознавание номера Х означает, что все байты с предыдущими номерами уже получены. Этот механизм позволяет регистрировать появление дубликатов в условиях повторной передачи. Нумерация байтов в пределах сегмента осуществляется так, чтобы первый байт данных сразу вслед за заголовком имел наименьший номер, а следующие за ним байты имели номера по возрастающей.

Окно, посылаемое с каждым сегментом, определяет диапазон номеров очереди, которые отправитель окна (он же получатель данных) готов принять в настоящее время. Предполагается, что такой механизм связан с наличием в данный момент места в буфере данных.

Варьируя величину окна, можно влиять на загрузку сети. Чем больше окно, тем большую порцию неподтвержденных данных можно послать в сеть. Но если пришло большее количество данных, чем может быть принято программой TCP, данные будут отброшены. Это приведет к излишним пересылкам информации и ненужному увеличению нагрузки на сеть и программу TCP.

С другой стороны, указание окна малого размера может ограничить передачу данных скоростью, которая определяется временем путешествия по сети каждого посылаемого сегмента. Чтобы избежать применения малых окон, получателю данных предлагается откладывать изменение окна до тех пор, пока свободное место не составит 20-40 % от максимально возможного объема памяти для этого соединения. Но и отправителю не стоит спешить с посылкой данных, пока окно не станет достаточно большим. Учитывая эти соображения, разработчики протокола TCP предложили схему, согласно которой при установлении соединения заявляется большое окно, но впоследствии его размер существенно уменьшается.

Если сеть не справляется с нагрузкой, то возникают очереди в промежуточных узлах - маршрутизаторах и в конечных узлах-компьютерах.

При переполнении приемного буфера конечного узла «перегруженный» протокол TCP, отправляя квитанцию, помещает в нее новый, уменьшенный размер окна. Если он совсем отказывается от приема, то в квитанции указывается окно нулевого размера. Однако даже после этого приложение может послать сообщение на отказавшийся от приема порт. Для этого сообщение должно сопровождаться пометкой «срочно». В такой ситуации порт обязан принять сегмент, даже если для этого придется вытеснить из буфера уже находящиеся там данные. После приема квитанции с нулевым значением окна протокол-отправитель время от времени делает контрольные попытки продолжить обмен данными. Если протокол-приемник уже готов принимать информацию, то в ответ на контрольный 'запрос он посылает квитанцию с указанием ненулевого размера окна.

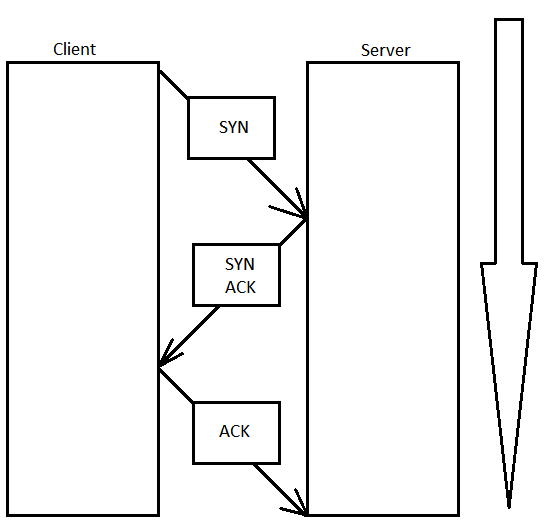
Другим проявлением перегрузки сети является переполнение буферов в маршрутизаторах. В таких случаях они могут централизованно изменить размер окна, посылая управляющие сообщения некоторым конечным узлам, что позволяет им дифференцированно управлять интенсивностью потока данных в разных частях сети.

16. Механизм установки TCP-соединения. Уязвимость TCP-протокола вида «парадокс дней рождения».

Механизм установки tcp-соединения

Выполняется с помощью трехкратного рукопожатия.

+Сервер прослушивает определенный порт на предмет запросов на соединение (пакет с флагом SYN). В ответ сервер посылает пакет с флагами SYN и ACK. Получив этот пакет, клиент отсылает в ответ пакет с флагом ACK.



Клиент посылает запрос на установление соединения на 80-ый порт на IP сервера.

Сервер узнает о IP и порте клиента из первого пакета (пакет с флагом SYN). Однако сервер не обязательно должен работать со стандартного порта, он может задать новые значения, передав их в пакете SYNACK. Чтобы не возникло путаницы, какой ответ идет на какой запрос (ведь может устанавливаться много соединений одновременно) используется значение поля «Порядковый номер» (т.е. при установке соединения поле используется особым образом, а именно для установления соответствия между пакетами (его значение генерируется случайным образом, и используется для сравнения)).

## Уязвимость tcp-протокола вида «парадокс дней рождения»

Если кто-то отправит пакет SYNACK с указанным своим IP и портом, и этот пакет дойдет до клиента раньше, чем SYNACFK от настоящего сервера, то соединение установится с узлом-злоумышленником. При этом, злоумышленник должен угадать значение поля «порядковый номер» (это 32 бита значений). Далее злоумышленник может установить соединение с реальным сервером и передавать сообщения между сервером и клиентом при этом прослушивая их, а следовательно завладевая информацией (пароли и т.д.) При этом с точки зрения клиента все будет работать нормально. Подобная атака называется «Man in the middle»

Произвольное значение IP и порта отправителя может быть задано только в пакетах типа ACKSYN. Это позволяет делать системы более гибкими.

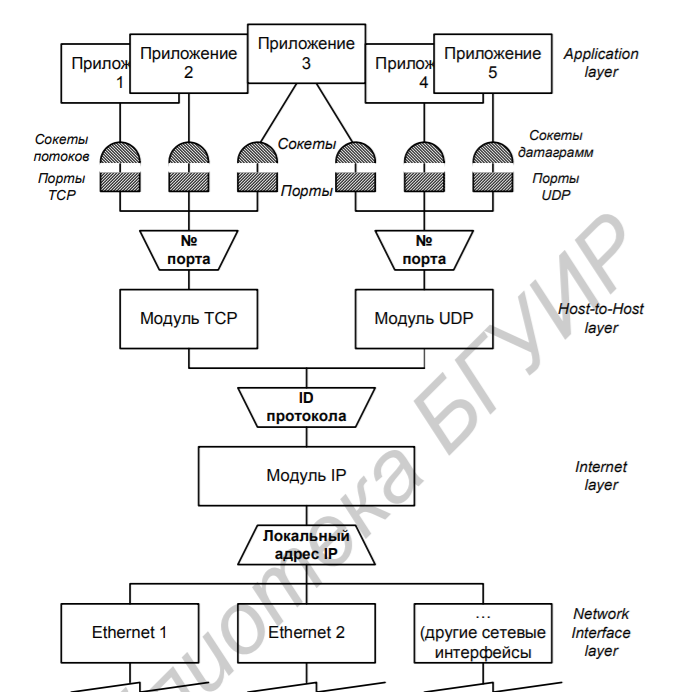
Конечно, что бы осуществить подобную атаку, злоумышленнику нужно угадать IP и порт клиента и значение «порядковый номер». Однако это не сложно: провайдеры используют определенные под диапазоны IP для районов. По поводу порядкового номера (4 байта) здесь срабатывает парадокс дня рождения:

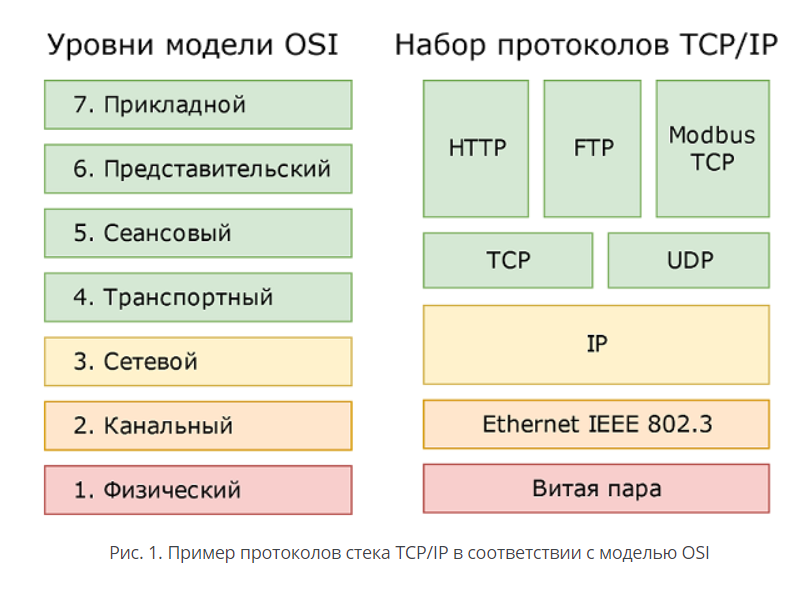
Пусть в комнате 366 человек. Вероятность того, что у двоих совпадут дни рождения 100% т.к. людей больше, чем дней. Но если сократить людей до 15, то вероятность будет в районе 70%, а для 50 человек она составит 99%. На первый взгляд это парадоксально. Однако вероятность растет за счет того, что происходит проверка «каждый с каждым» т.е. полное декартово произведение.

Таким образом, вероятность угадать чей-нибудь «порядковый номер» довольно высока. Значит, злоумышленник не может прослушать конкретного пользователя, но может прослушать кого-нибудь (кого повезет). Данная уязвимость устраняется только в IPv6.

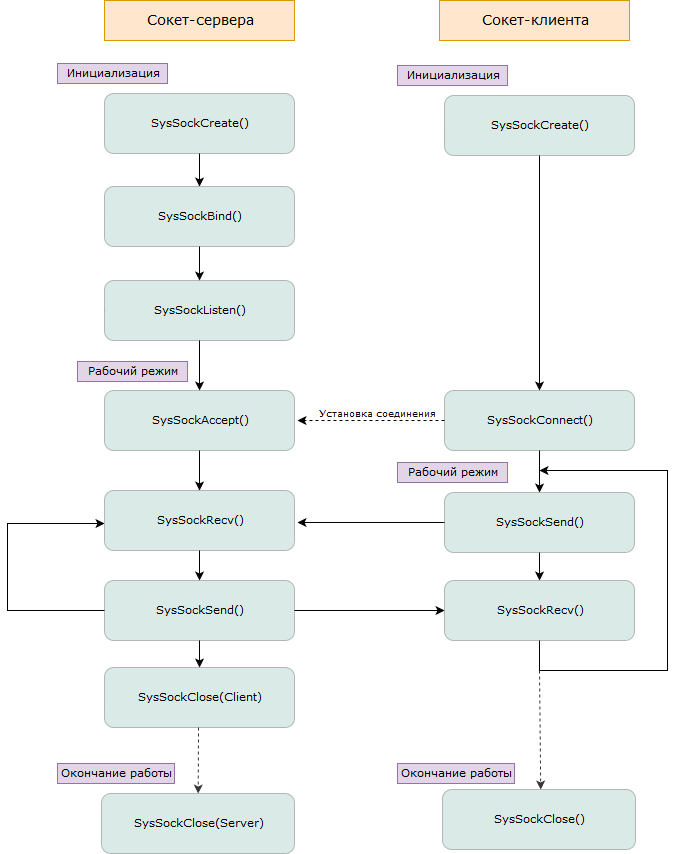
17. Интерфейс прикладного программирования Sockets («сетевые гнезда»). Взаимодействие клиента с сервером по протоколам UDP и TCP с помощью прикладного интерфейса Sockets.

Сокет (socket, букв. гнездо, соединитель) – программный объект, обычно системный, абстрагирующий точку доступа к транспортной системе. В типичном случае сокет сопоставляется одному из портов одного из транспортных протоколов и служит удобной для прикладного программирования унифицированной надстройкой над ним. Реже используются сокеты, связанные с другими протоколами, либо «сырые».



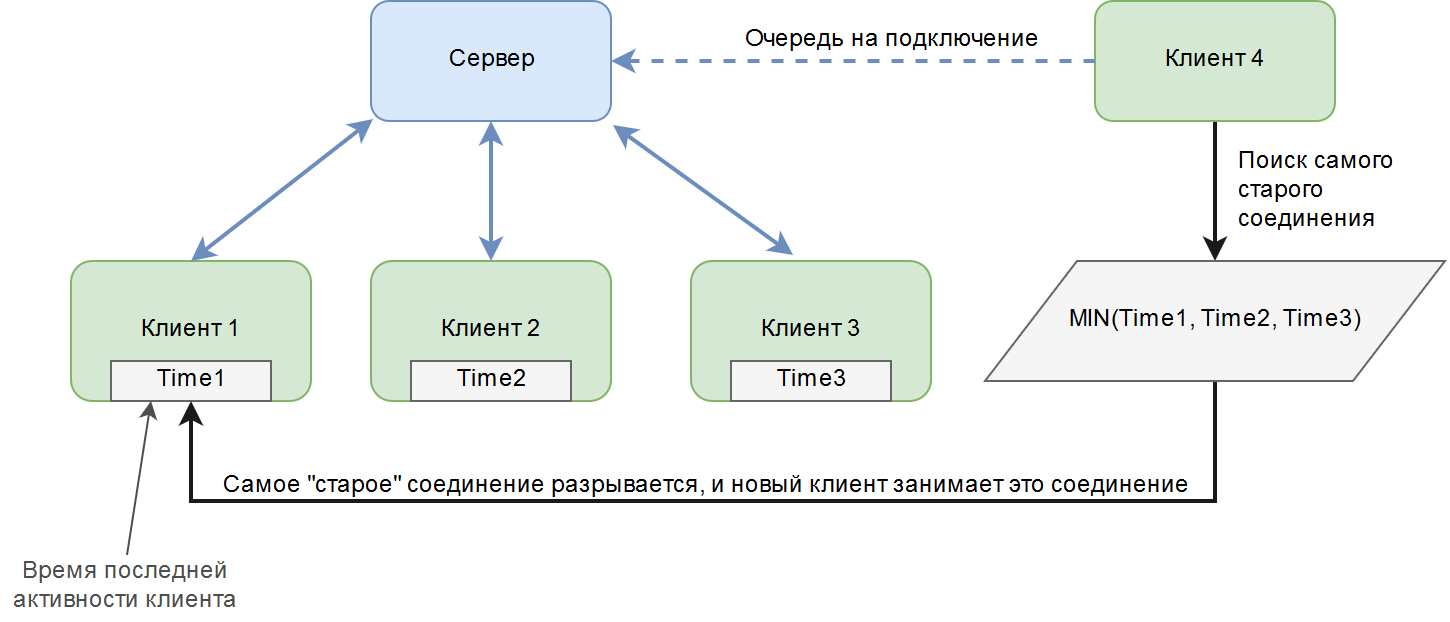
Сокеты в рассматриваемом здесь виде появились в BSD Unix, в настоящее время они входят в стандарт POSIX и являются практически общепринятым интерфейсом межпроцессного взаимодействия, в первую очередь в сетях. Их поддержка в различных системах может осуществляться непосредственно ядром как часть подсистемы ввода-вывода, либо специальными библиотеками. Благодаря стандартизации принципы программирования с использованием сокетов в различных ОС в достаточной мере подобны, хотя и имеют некоторые особенности. Для прикладных программ сокеты обычно доступны через их описатели (descriptor, handle). Часто описатели сокетов совместимы с описателями файлов, что позволяет унифицировать процедуры ввода-вывода. В целом API сокетов позволяет реализовать все задачи межпроцессного взаимодействия: идентификацию участников, синхронизацию, взаимное исключение, обмен данными с установлением соединения (посредством виртуального канала) и без него (пакетная, или датаграммная, передача), а также низкоуровневое (прямое) управление интерфейсом. 

TCP сокеты используют TCP-соединения, в которых на транспортном уровне (рис. 1) обеспечивается надёжная доставка данных. TCP протокол отвечает за установление и поддержание соединения, сегментацию, доставку и буферизацию данных, упорядочивание и избавление от дублированных TCP-сегментов данных, контроль ошибок и скорости передачи. Схема работы простого TCP сокета представлена на следующей схеме.



После успешных приема и передачи данных может быть реализовано несколько вариантов поведения программы:

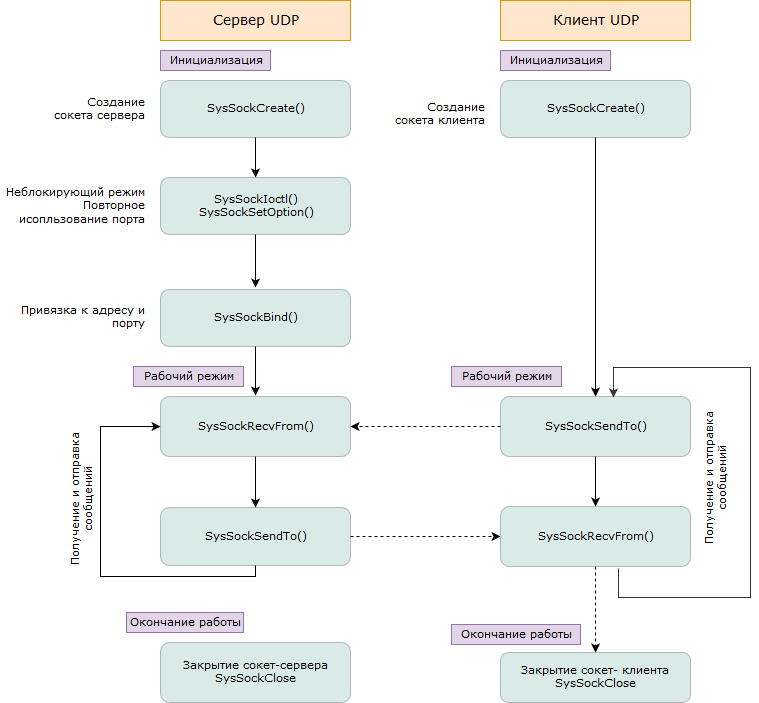
* 1. Программа может закрыть клиентское соединение. В таком случае, в следующих циклах программы сервер будет ожидать подключения с новым клиентом. Такой режим работы не является эффективным, так как контроллеру придется во время каждого цикла закрывать клиентское соединение и подключать нового (или того же самого что и в предыдущем цикле) клиента.
* 2. Программа может не закрывать клиентский сокет, а сохранить установленное соединение. В таком случае, один раз установив соединение, клиент будет постоянно отправлять и получать данные от сервера. Такой режим работы более эффективный, но может возникнуть ситуация, когда все клиентские соединения будут заняты, и новый клиент не сможет подключиться к серверу. Решить данную ситуацию можно различными способами. Один из вариантов – следить за последним временем активности клиентских сокетов, и отключать самое старое соединение в случае, если в очереди обнаружился новый клиент (следующая схема)



**Особенности сокетов TCP**

Использование TCP сокетов позволяет приложениям клиента и сервера обмениваться данными почти прозрачно, не заботясь о поддержании сетевого соединения, доставке пакетов по сети, порядке передачи пакетов и буферизации. TCP сокеты гарантируют доставку сообщений и правильный порядок пакетов, а также пересылают пакеты повторно, если подтверждение о передаче не приходит в течение определенного промежутка времени. Таким образом, использовать TCP сокеты уместно там, где необходима гарантированная доставка данных сетевыми средствами.  
  
Несмотря на многие преимущества, TCP сокеты имеют и негативные стороны. Например, необходимость поддержания TCP-соединения уменьшает пропускную способность обмена данными в распределенных системах. Также, в системах обмена данными реального времени повторная передача потерянных пакетов может привести к тому, что система получит данные, которые утратили свою актуальность.

**UDP сокеты**

Все перечисленные недостатки TCP сокетов связаны с особенностью TCP-протокола. Если в системе присутствие данных факторов крайне нежелательно, а гарантированность доставки сообщений не является критичным требованием, то в качестве альтернативы TCP сокетов могут использоваться UDP (датаграммные) сокеты.  
  
UDP сокеты устроены проще, чем TCP. В качестве транспортного уровня используется протокол UDP, который не требует установления соединения и подтверждения приема. Информация пересылается в предположении, что принимающая сторона ее ожидает. Датаграммные сокеты не контролирует ничего, кроме целостности полученных датаграмм. Несмотря на это, UDP сокеты нашли свое применение в системах, где на первом месте стоит именно актуальность данных и их быстрая доставка, а не гарантия доставки каждого сообщения.  
  
Например, сервер в ответ на запросы клиента передает по сети текущие (мгновенные) значения некоторого параметра контролируемого технологического процесса, а клиент формирует управляющий сигнал на основе принятых значений. Если темп опроса сервера клиентом много больше требуемого времени реакции алгоритма управления на изменение значения контролируемого параметра, то потеря одного-двух сообщений от сервера несущественно повлияет на качество формирования управляющего сигнала. В случае использования TCP соединения потерянное сообщение будет автоматически передано повторно, что может привести к получению клиентом неактуального значения контролируемого параметра и к формированию неправильного управляющего сигнала.  


18. Виды узлов сети. Усилитель, повторитель, коммутатор, маршрутизатор, мост, шлюз.

Созданная на определенном этапе развития фирмы локальная вычислительная сеть с течением времени перестает удовлетворять потребности всех пользователей и воз­никает необходимость расширения ее функциональных возможностей или границ охватываемой ею территории. Может возникнуть необходимость объединения внутри фирмы ЛВС различных отделов и филиалов для организации обмена дан­ными.

Наконец, стремление получить выход на новые информационные ресурсы может потребовать подключения ЛВС к сетям более высокого уровня. В качестве межсетевого интерфейса для соединения сетей между собой используются:  
 -   повторители;  
 -   мосты;

 -   маршрутизаторы;  
 -   шлюзы.  
**Повторители***(repeater) —*устройства, усиливающие электрические сигналы и обес­печивающие сохранение формы и амплитуды сигнала при передаче его на боль­шие расстояния. Повторители выполняют лишь регенерацию паке­тов данных, обеспечивая тем самым электрическую независимость сопрягаемых сетей и защиту сигналов от воздействия помех. Использование усилителей позво­ляет расширить и протяженность одной сети, объединяя несколько сегментов сети в единое целое. При установке усилителя создается физический разрыв в линии связи, при этом сигнал воспринимается с одной стороны, регенерируется и направ­ляется к другой части линии связи.  
**Мосты***(bridge) —*регулируют тра­фик (передачу данных) между сетями, использующими одинаковые протоколы передачи данных на сетевом и выше уровнях, выполняя фильтрацию информаци­онных пакетов в соответствии с адресами получателей. Мост может соединять сети разных топологий, но работающие под управлением однотипных сетевых опера­ционных систем.  
**Маршрутизаторы** *(router) —*обеспечивают соединение логически не связанных се­тей; они анали­зируют сообщение, определяют его дальнейший наилучший путь, выполняют его некоторое протокольное преобразование для согласования и передачи в другую сеть, создают нужный логический канал и передают сообщение по назначению. Маршру­тизаторы обеспечивают достаточно сложный уровень сервиса: они могут, например, соединять сети с разными методами доступа; могут перераспределять нагрузки в ли­ниях связи, направляя сообщения в обход наиболее загруженных линий и т. д.  
**Шлюзы** *(gateway) —*устройства, позволяющие объединить вычислительные сети, использующие различные протоколы OSI на всех ее уровнях; они выполняют про­токольное преобразование для всех семи уровней управления модели OSI. Кроме функций маршрутизаторов они выполняют еще и преобразование формата инфор­мационных пакетов и их перекодирование, что особенно важно при объединении неоднородных сетей.  
Мосты, маршрутизаторы и шлюзы в локальной вычислительной сети — это, как правило, выделенные компьютеры со специальным программным обеспечением и дополнительной связной аппаратурой.

**Методы коммутации:**  
Основным назначением узлов коммутации является прием, анализ, а в сетях с маршрутизацией еще и выбор маршрута, и отправление данных по выбранному направлению. В общем случае узлы коммутации включают в себя и устройства межсетевого интерфейса.  
Узлы коммутации вычислительных сетей содержат устройства коммутации *(коммутаторы).*Если они выполняют коммутацию на основе иерархических сетевых адресов, их называют *маршрутизаторами.*  
*Устройства коммутации*занимают важное место в системах передачи информации в вычислительных сетях. С помощью устройств коммутации значительно сокращается протяженность каналов связи в сетях с несколькими взаимодействующими абонентами: вместо того чтобы прокладывать несколько каналов связи от данного абонента ко всем остальным, можно проложить лишь по одному каналу от каждого абонента к общему коммутационному узлу. Узлы коммутации осуществляют один из трех возможных видов коммутации при передаче данных:

* коммутацию каналов;
* коммутацию сообщений;
* коммутацию пакетов.

Коммутация каналов  
Между пунктами отправления и назначения устанавливается непосредственное физическое соединение путем формирования составного канала из последовательно соединенных отдельных участков каналов связи. Такой сквозной физический составной канал организуется в начале сеанса связи, поддерживается в течение всего сеанса и разрывается после окончания передачи.  
Недостатки:

* большое время создания канала
* снижение общей пропускной способности сети
* физический канал часто бывает недогружен.

Применяется этот метод коммутации чаще всего при дуплексной передаче аудиоинформации (обычная телефонная связь — типичный пример коммутации каналов).  
Коммутация сообщений  
Данные передаются в виде дискретных порций разной длины (сообщений), причем между источником и адресатом сквозной физический канал не устанавливается и ресурсы коммуникационной системы предварительно не распределяются. Отправитель лишь указывает адрес получателя. Узлы коммутации анализируют адрес и текущую занятость каналов и передают сообщение по свободному в данный момент каналу на ближайший узел сети в сторону получателя. Применяется этот вид коммутации в электронной почте, телеконференциях, электронных новостях и т. п.

**Коммутация пакетов**  
В современных системах для повышения оперативности, надежности передачи и уменьшения емкости запоминающих устройств узлов коммутации длинные сообщения разделяются на несколько более коротких стандартной длины, называемых *пакетами*(иногда очень короткие сообщения, наоборот, объединяются вместе в пакет). Стандартность размера пакетов обусловливает соответствующую стандартную разрядность оборудования узлов связи и максимальную эффективность его использования. Пакеты могут следовать к получателю даже разными путями и непосредственно перед выдачей абоненту объединяются (разделяются) для формирования законченных сообщений. Этот вид коммутации обеспечивает наибольшую пропускную способность сети и наименьшую задержку при передаче данных. Недостатком коммутации пакетов является трудность его использования для систем, работающих в интерактивном режиме и в реальном масштабе времени.

**Методы маршрутизации:**  
В сетях с маршрутизацией информации возникает задача маршрутизации данных. В системах с коммутацией каналов и при создании виртуального канала маршрутизация выполняется один раз при установлении начального соединения. При обычных режимах коммутации пакетов и сообщений маршрутизация выполняется непрерывно, по мере прохождения данных от одного узла коммутации к другому.  
**Маршрутизация заключается в правильном выборе выходного канала в узле коммутации на основании адреса, содержащегося в заголовке пакета (сообщения).**

* *Простая маршрутизация*при выборе дальнейшего пути для сообщения (пакета) учитывает лишь статическое состояние сети, ее текущее состояние —загрузка и изменение топологии из-за отказов — не учитывается. Одно из направлений простой маршрутизации — лавинное отправление сообщения сразу по всем свободным каналам. О достоинствах такой маршрутизации говорить не приходится.
* *Фиксированная маршрутизация*учитывает только изменение топологии сети. Для каждого узла назначения канал передачи выбирается по электронной *таблице маршрутов*(route table), определяющей кратчайшие пути и время доставки информации до пункта назначения. Эта маршрутизация используется в сетях с установившейся топологией.
* *Адаптивная маршрутизация*учитывает и изменение загрузки, и изменение топологии сети. При выборе маршрута информация из таблицы маршрутов дополняется данными о работоспособности и занятости каналов связи, оперативнойинформацией о существующей очереди пакетов на каждом канале. В локальном варианте этой маршрутизации учитываются данные только о каналах, исходящих из текущего узла, а при распределенной адаптивной маршрутизации и данные, получаемые от соседних узлов коммутации.

**Способы повышения производительности ЛВС**  
Используются три основных способа повышения производительности сети:

* выбор высокоскоростных технологий передачи данных;
* сегментация структуры сети;
* использование технологии коммутации кадров.

**Устранение неисправностей в сети**  
Большинство нарушения сетевых соединений кратковременны и связаны с работами администратора по перезапуску, переустановке и изменению настроек устройств, отвечающих за доставку запросов и сообщений. Есть всего несколько условий, нарушение которых не позволяет соединиться с сервером почты, Web-сервером и т.п.:

* сервер должен нормально функционировать;
* должен существовать путь между двумя компьютерами;
* компьютер должен уметь разрешать имя DNS (определять IP-адрес по имени компьютера);
* необходимо иметь соответствующие права доступа.

Чтобы отыскать причину неполадок, необходимо:

* Проверить конфигурацию протокола TCP/IP на компьютере;
* Убедиться, что имя удаленного компьютера верно и имеет корректный адрес TCP/IP;
* Убедиться, что доступ к удаленному компьютеру возможен. Если в результате подтверждает правильность перевода имени DNS в адрес TCP/IP, работоспособность удаленного компьютера и наличие открытого пути между двумя системами, то проблемы, скорее всего, связаны с идентификацией и правами доступа либо с работой конкретной службы на удалённом компьютере.

Утилиты командной строки для проверки соединений

  Команда **Ipconfig /all** показывает настройки протокола TCP/IP на компьютере. Два важных параметра протокола - Default Gateway (шлюз по умолчанию) и сервер DNS.  
    
Команда **Nslookup** переводит TCP/IP-имя сервера в TCP/IP-адрес. Эта команда аналогична использованию он-лайновой телефонной книжки для поиска индивидуального имени (TCP/IP-имя) и телефонного номера (TCP/IP-адрес). По умолчанию, если набрать **Nslookup** в командной строке, система отправляет запрос к серверу DNS, указанному в установках протокола TCP/IP на компьютере. Если сервер DNS не работает, появляется сообщение об ошибке: *DNS request timed out*. Если все в порядке, сервер DNS отвечает символом «больше» (**>**) и ожидает ввода имени сервера, который необходимо найти. Когда сервер DNS возвращает имя и адрес искомого сервера, в их корректности можно не сомневаться.  
    
Команда **Ping “компьютер”** дает информацию о доступности и состоянии удаленного компьютера. Команда **Ping** работает и по имени, и по адресу. Если при выполнении команды **Ping** для компьютера **www.google.com** пришел ответ на запрос, значит, для имени найден правильный TCP/IP-адрес, и компьютер **Google** функционирует нормально. Неработающий компьютер не отвечает на запрос **Ping** и появляется сообщение об ошибке: *Request timed out*.  
Возможны три причины ошибки:

* неверное имя;
* сервер DNS не может определить адрес TCP/IP;
* необходимые службы или сам компьютер не работают.

Кроме того, операционная система может не отвечать на запросы**Ping**, если администратор специально заблокировал эту возможность, например, из соображений безопасности.

Команда **Tracert** показывает все промежуточные узлы между источником и приемником. **Tracert**, как и **Ping,**работает и по имени, и по TCP/IP-адресу. Использование **Tracert** – это простой способ проверить, есть ли для запроса открытый путь к получателю. **Tracert** отвечает списком узлов, одна строка – один узел, который необходимо пройти при маршрутизации запроса. Если один из узлов на пути запроса недоступен, команда возвращает временные параметры и имя узла, отмеченные символом звездочка (\*), и сообщение об ошибке *Request timed out*. Если проложить маршрут до получателя невозможно, необходимо подождать, пока отсутствующая часть не заработает, либо не появится альтернативный путь. Первым узлом, который покажет **Tracert**, должен быть Default Gateway (шлюз по умолчанию), установленный в настройках протокола TCP/IP компьютера.