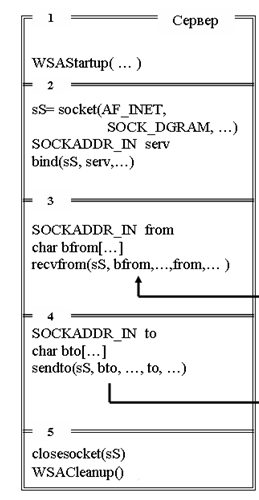
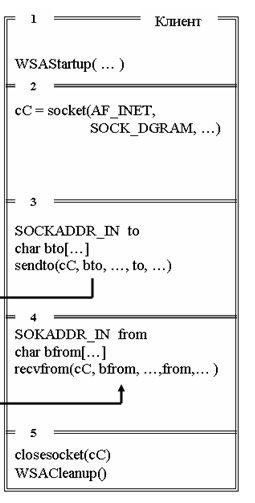
1. Структура UDP-сервера.

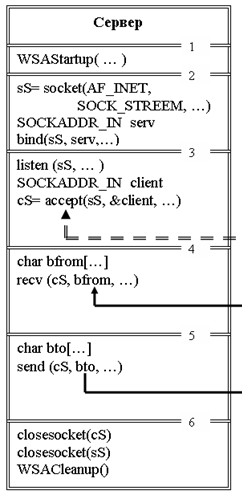


1. Структура UDP-клиента.



1. Отличие UDP-сервера от TCP-сервера.

TCP сервер:



Между сокетами курсируют UDP-пакеты, и поэтому вся работа, связанная с обеспечением надежности и установкой правильной последовательности передаваемых пакетов возлагается на само приложение.

Создание сокета: указываем параметр SOCK\_DGRAM – сокет, ориентированный на сообщения.

3 блок в TCP-сервере: создание соединения (listen и accept). В udp блок отсутствует, после связки сокета, сервер сразу начинает ждать поступления данных от клиента.

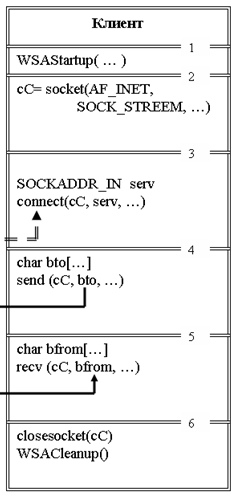
Используются функции recvfrom и sendto вместо recv и send

UDP — это **протокол без установления соединения** (connectionless). Сервер и клиент просто обмениваются сообщениями (датаграммами) без установки постоянного соединения. UDP-сокет может принимать сообщения от нескольких клиентов через один и тот же сокет.

* **TCP**: Адресация и установление соединения происходит при вызове connect(). После этого все данные идут по одному маршруту, и адрес отправителя не нужно каждый раз указывать.
* **UDP**: Адрес отправителя не связан с сокетом напрямую, и каждый пакет может приходить от разных источников, поэтому для получения данных требуется передавать информацию об адресе отправителя.

1. Отличие UDP-клиента от TCP-клиента.

TCP-клиента:



Создание сокета: указываем параметр SOCK\_DGRAM – сокет, ориентированный на сообщения.

Нет необходимости устанавливать соединение. Данные могут быть отправлены сразу после создания сокета (отсутствует connect).

Используются функции recvfrom и sendto вместо recv и send

1. Что такое канал связи с точки зрения TCP?

Канал связи в контексте **TCP (Transmission Control Protocol)** — это абстракция, которая обозначает логическое соединение между двумя узлами (обычно компьютерами) для обмена данными.

Канал связи (или соединение) создается между двумя сокетами, ориентированными на поток. На стороне сервера это должен быть связанный (функция bind) и переключенный в режим прослушивания (функция listen) сокет. На стороне клиента должен быть создан дескриптор ориентированного на поток сокета (функция socket).

Канал связи создается в результате взаимодействия функций accept (на стороне сервера) и connect (на стороне клиента). Алгоритм взаимодействия этих функций зависит от установленного режима ввода-вывода для участвующих в создании канала сокетов.

Обмен данными по каналу связи осуществляется между двумя сокетами и возможен сразу после того, как этот канал создан. Функция send пересылает по каналу, указанного сокета, определенное количество байт данных. Функция recv принимает по каналу, указанного сокета, определенное количество байт данных.

1. Что значит: обмен без соединения(ориентированный на сообщения)?

**Обмен без соединения (ориентированный на сообщения)** означает способ передачи данных в сетях, при котором не требуется устанавливать постоянное соединение между отправителем и получателем для передачи данных.

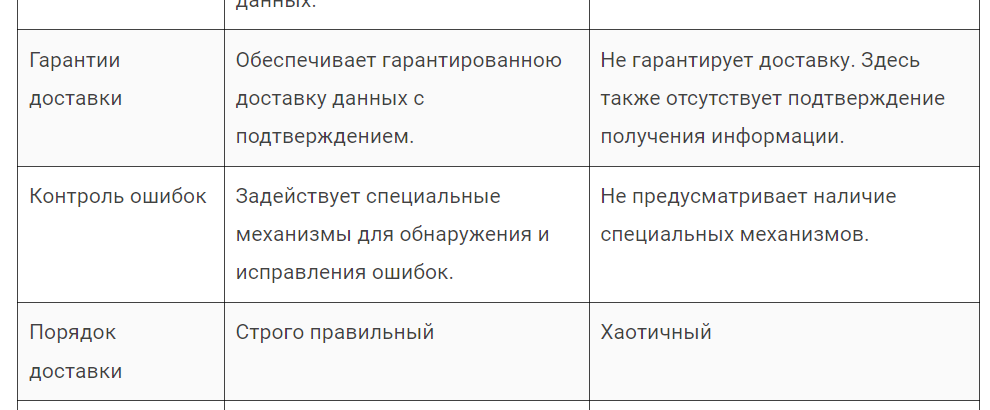
**Отсутствие установки соединения**: При обмене без соединения отправитель просто отправляет сообщение (датаграмму), не устанавливая предварительно соединение с получателем. Нет необходимости в процессе "рукопожатия", как в TCP.

**Ориентация на сообщения**: Данные отправляются как отдельные блоки (сообщения или датаграммы). Каждое сообщение является самодостаточной единицей данных, и границы сообщения явно определены. Это отличается от **потоковой** передачи данных, когда данные передаются в непрерывном потоке байтов (как в TCP).

**Нет гарантии доставки**: В отличие от TCP, который обеспечивает надежную доставку данных, при обмене без соединения сообщения могут теряться, дублироваться или приходить в неправильном порядке. Протокол сам по себе не обеспечивает механизмов для повторной отправки потерянных данных или восстановления правильной последовательности.

1. Что значит UDP ненадежный протокол?

Особенностью использования этих функций заключается в том, что протоколом UDP не гарантируется доставка и правильная последовательность приема отправленных сообщений. Весь контроль надежности доставки и правильной последовательностью поступления сообщений возлагается на разработчика приложения.



1. В каких случаях следует выбирать UDP-протокол для обмена данными?

В связи с тем, что UDP – ненадёжный протокол, обмен данными с помощью сообщений используется, в основном, для широковещательных сообщений (UDP не требует установки соединения с каждым отдельным получателем, что упрощает отправку сообщения сразу нескольким устройствам) или для пересылки коротких сообщений, последовательность получения которых не имеет значения.

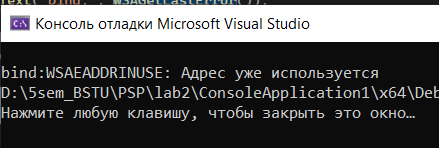
Чаще всего UDP применяется в чувствительных ко времени службах, где потерять пакеты лучше, чем ждать.

1. Что значит: UDP не буферизирует соединения?

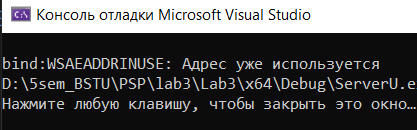
|  |  |
| --- | --- |
| TCP | UDP |
| Вычисление и проверка контрольной суммы. Перед отправкой протокол TCP вычисляет контрольную сумму, а протокол TCP получателя проверяет ее. Если контрольная сумма неправильная, получатель не подтверждает получение пакета. | Вычисление и проверка контрольной суммы. Перед отправкой протокол UDP вычисляет контрольную сумму, а протокол UDP принимающей стороны проверяет ее. Пакеты с неправильной контрольной суммой отбрасываются. |
| TCP обнаруживает ошибки в контрольной сумме, а также потерянные пакеты и фрагменты по тайм-ауту. При возникновении ошибки TCP передает данные повторно до тех пор, пока они не будут успешно приняты (или сообщает приложению о неисправимой ошибке). | UDP не отправляет подтверждение о приеме пакетов и не обнаруживает их потери из-за ошибок передачи или переполнения пула буферов. Поэтому UDP никогда не выполняет повторной передачи пакета. Исправление ошибок должно выполняться приложением. |
| работает не с последовательностью записей, а с потоком байтов. Если буфер передачи содержит больше данных, чем может обработать приемник, они разбиваются на части в соответствие с размером MTU. Так как недостаток места в буфере обрабатывается неявно, TCP не гарантирует, что число и размер операций приема будет совпадать с числом и размером операций передачи. Разбиение потока на записи и блоки должно выполняться приложением. | Размер данных не должен превышать размер буфера. Это означает, что размеры пулов буферов в обеих системах должны быть установлены в соответствии с требованиями приложений. Максимальный размер пакета UDP составляет 64 Кб. Если приложение отправляет блоки данных большего размера, оно может самостоятельно разбивать их на дейтаграммы (примером может служить DCE), либо воспользоваться протоколом TCP. |

1. Можно ли одновременно задействовать один номер порта для двух серверов: TCP и TCP, UDP и UDP, TCP и UDP? Если возможно, то продемонстрируйте.

TCP и TCP – нельзя



UDP и UDP – нельзя (сразу же срабатывает ошибка).



TCP и UDP – можно. Сервер выбирается в зависимости от Sock\_Dgram/Sock\_Stream клиента

1. Определите порт UDP-сервера с помощью утилиты NETSTAT.
2. Продемонстрируйте потерю сообщений UDP-сервером. Используйте функцию Sleep.