Лабораторная работа 10

ПСКП

**Задание 01**

1. Разработайте приложение **10-01**, представляющее собой TCP-сервер. Сервер должен через TCP-соединение принимать строковое сообщение от TCP-клиента и возвращать клиенту текст полученного сообщения с префиксом **ECHO:**.
2. Разработайте приложение **10-01a**, представляющее собой TCP-клиента, проверяющего работоспособность сервера 10-01.
3. Разработайте TCP-сервер на языке С++, выполняющий те же функции, что и сервер 10-01. Проверьте его работоспособность с клиентом 10-01a.
4. Разработайте TCP-клиента на языке С++, выполняющего те же функции, что и клиент 10-01a. Проверьте его работоспособность с сервером 10-01.

**Задание 02**

1. Разработайте приложение **10-02**, представляющее собой TCP-сервер, прослушивающий 2 порта: 40000, 50000. Сервер должен через TCP-соединение принимать поток 32-битовых чисел (по одному числу за каждую отправку клиентом). Сервер суммирует полученные числа и каждые 5 сек. отправляет клиенту полученную (промежуточную) сумму. Сервер обеспечивает каждому подключенному клиенту получение правильных промежуточных сумм чисел, отправленных клиентом серверу. Сервер должен обеспечивать вывод на консоль диагностических сообщений, позволяющих проверить корректность его работы.
2. Разработайте приложение **10-02a**, представляющее собой TCP-клиента. Клиент принимает 2 числовых параметра (номер порта сервера и число **X**) через командную строку. Клиент через TCP-соединение отправляет 1 раз в секунду серверу 32-битовое число **X**. Клиент принимает от сервера промежуточные суммы и выводит их на консоль.
3. Проверьте работоспособность сервера с двумя клиентами, устанавливающими соединение с сервером через разные порты и отправляющими разные числа.
4. Проверьте работоспособность сервера с тремя клиентами, устанавливающими соединение с сервером через разные порты и отправляющими разные числа.

**Задание 03**

1. Разработайте приложение **10-03**, представляющее собой UDP-сервер. Сервер должен принимать строковое сообщения от UDP-клиента и возвращать клиенту текст полученного сообщения с префиксом **ECHO:**.
2. Разработайте приложение **10-03a**, представляющее собой UDP-клиента, проверяющего работоспособность сервера 10-03.
3. Разработайте UDP-сервер на языке С++, выполняющий те же функции, что и сервер 10-03. Проверьте его работоспособность с клиентом 10-03a.
4. Разработайте UDP-клиента на языке С++, выполняющего те же функции, что и клиент 10-03a. Проверьте его работоспособность с сервером 10-03.

**Задание 04.** Ответьте на следующие вопросы.

1. Поясните основные свойства протокола TCP.
2. Поясните процедуры установки и закрытия TCP-соединения.
3. Поясните понятие «порт».
4. Поясните понятие «сокет».
5. Поясните понятие «полудуплексный канал связи».
6. Поясните понятие «дуплексный канал связи».
7. Поясните отличие протокола UDP от TCP.

**13. Основные свойства протокола TCP:**

* **Надежность передачи данных:** TCP обеспечивает надежную передачу данных, используя механизм подтверждений и повторной передачи в случае потери пакетов.
* **Управление потоком данных:** TCP регулирует поток данных между отправителем и получателем, чтобы избежать перегрузок или переполнений буфера.
* **Установка соединения:** Протокол TCP устанавливает соединение между двумя конечными точками (хостами) перед передачей данных.
* **Полный дуплекс:** TCP поддерживает полнодуплексную связь, что означает, что обе стороны могут одновременно отправлять и принимать данные.
* **Управление порядком данных:** Данные, передаваемые по TCP, приходят в порядке, в котором они были отправлены.

**14. Процедуры установки и закрытия TCP-соединения:**

* **Установка соединения:**
  1. **Шаг начала соединения (Three-Way Handshake):** Клиент отправляет сегмент SYN (синхронизация) серверу. Сервер отвечает сегментом SYN-ACK. Клиент отправляет подтверждающий сегмент ACK.
  2. **Установление соединения:** Теперь соединение установлено, и данные могут быть переданы.
* **Закрытие соединения:**
  1. **Шаг начала закрытия:** Одна из сторон отправляет сегмент FIN (финиш) для начала процесса закрытия.
  2. **Ответное закрытие:** Получив сегмент FIN, вторая сторона отправляет подтверждающий сегмент ACK, и происходит закрытие направления передачи данных.
  3. **Завершение соединения:** После завершения передачи данных в обоих направлениях, каждая сторона отправляет сегмент FIN, и после получения подтверждения ACK, соединение считается закрытым.

**15. Понятие «порт»:**

Порт - это логическое числовое обозначение, используемое для идентификации конкретного процесса или службы в компьютерной сети. Порты делятся на два диапазона: порты от 0 до 1023 известны как "привилегированные" или "зарезервированные" порты, а порты от 1024 до 65535 известны как "непривилегированные" или "динамические" порты.

**16. Понятие «сокет»:**

Сокет - это программный интерфейс для обеспечения обмена данными между процессами на разных узлах сети. Сокет состоит из IP-адреса и порта. Это комбинация IP-адреса и номера порта, который создает точку связи для обмена данными между двумя процессами через сеть.

**17. Понятие «полудуплексный канал связи»:**

Полудуплексный канал связи - это канал, который позволяет передачу данных в обе стороны, но не одновременно. Поочередная передача данных может осуществляться только в одном направлении.

**18. Понятие «дуплексный канал связи»:**

Дуплексный канал связи - это канал, который позволяет одновременную передачу данных в обе стороны. Оба конца канала могут одновременно отправлять и принимать данные.

**19. Отличие протокола UDP от TCP:**

* **Надежность:**
  + **TCP:** Обеспечивает надежную передачу данных с установкой соединения, подтверждением и повторной передачей. Гарантирует доставку данных в порядке, в котором они были отправлены.
  + **UDP:** Передает данные без гарантий надежной доставки, повторной передачи или подтверждения. Может быть быстрее, но менее надежен.
* **Управление потоком:**
  + **TCP:** Обеспечивает управление потоком данных для предотвращения переполнения буфера и перегрузок.
  + **UDP:** Не имеет механизма управления потоком; приложения должны сами заботиться о контроле потока.
* **Установка соединения:**
  + **TCP:** Имеет механизм установки и закрытия соединения (трехсторонний рукопожатий).
  + **UDP:** Не требует установки соединения; каждый пакет рассматривается независимо.
* **Заголовок пакета:**
  + **TCP:** Имеет более сложный заголовок, что может привести к большему объему данных.
  + **UDP:** Имеет более легкий заголовок, что может сэкономить пропускную способность сети.

Выбор между TCP и UDP зависит от требований конкретного приложения и уровня надежности, необходимого для передачи данных.

Создается двунаправленный поток (**duplex**), который связан с WebSocket-соединением. Затем читаемый поток (**file**) создается для файла 'client.txt', и его данные передаются через WebSocket на сервер.

При подключении клиента создается двунаправленный поток (**duplex**), который позволяет передавать данные в обе стороны. Затем читаемый поток (**file**) создается для файла 'client.txt', и его данные передаются через WebSocket на сервер. Затем создается поток записи (**downloaded**), в который попадают данные от клиента через WebSocket. Файл сохраняется локально, и выводится сообщение о завершении передачи.