Лаба 4

Теоретический конспект к лабе

1. **Широковещательные IP-адреса (Broadcast):**

Широковещательный адрес — это специальный IP-адрес, который используется для отправки сообщения всем устройствам в локальной сети. В IPv4 широковещательный адрес имеет вид, где все биты хостовой части адреса установлены в 1. Например, в сети с адресом 192.168.1.0/24 широковещательный адрес будет 192.168.1.255.

Широковещательная передача позволяет отправить сообщение сразу всем устройствам сети, что полезно, когда клиент или сервер должен обнаружить все доступные узлы в сети без явного указания их IP-адресов.

Применение в работе:

Клиент отправляет широковещательный запрос на заранее известный порт всем устройствам в сети, чтобы найти доступные серверы.

Серверы, настроенные на прием таких запросов, могут ответить клиенту.

2. **Основы работы с сокетами:**

Сокет — это программный интерфейс, через который приложения могут обмениваться данными по сети. В основе работы с сокетами в протоколах UDP и TCP лежат следующие функции:

– **socket()** — создает сокет для сетевого взаимодействия. Это начальная точка работы с сетями. В этой лабораторной работе используются сокеты типа **SOCK_DGRAM**, работающие по протоколу UDP (недоступные соединения).

– **bind()** — связывает сокет с конкретным IP-адресом и портом. Для серверов важно закрепить сокет за определенным портом, чтобы он мог принимать запросы на этот порт.

– **recvfrom()** — функция для приема данных через сокет. Эта функция используется для получения данных от клиента, когда сервер ожидает запроса.

В случае удачного приема данных возвращает количество полученных байтов.

В случае ошибки возвращает **SOCKET_ERROR** и код ошибки можно получить через **WSAGetLastError()**.

sendto() — функция для отправки данных через сокет. В данной работе она используется для отправки ответов клиенту.

Отправляет данные на определенный адрес и порт (клиента).

Используется как клиентом для отправки широковещательного запроса, так и сервером для ответа клиентам.

setsockopt() — функция, которая используется для изменения параметров работы сокета. В данном случае используется для включения широковещательной передачи с помощью опции **SO_BROADCAST**.

Применение в работе:

Клиент настраивает сокет для отправки широковещательных запросов с помощью **setsockopt**.

Сервер принимает и обрабатывает запросы клиентов через **recvfrom**, а затем отправляет ответ через **sendto**.

3. **Протокол UDP:**

UDP (User Datagram Protocol) — это протокол транспортного уровня, который работает без установления соединения. Это означает, что отправитель просто передает данные без подтверждения получения.

Преимущества UDP:

Быстрая передача данных, так как не требуется установка соединения.

Поддержка широковещательных и многоадресных (**multicast**) сообщений.

Недостатки UDP:

Нет гарантии доставки сообщений (пакеты могут быть потеряны или доставлены в неправильном порядке).

Нет механизма подтверждения получения (как в TCP).

Применение в работе:

UDP используется для быстрой передачи запросов и ответов между клиентом и сервером. Клиент отправляет широковещательный запрос на определенный порт, сервер его получает и отвечает на тот же порт.

4. **Ошибки при работе с сетевыми сокетами:**

В сетевых приложениях важно обрабатывать ошибки, возникающие при работе с сокетами. Например, ошибки при получении или отправке данных через сокет.

Основные ошибки и их обработка:

WSAETIMEDOUT — ошибка, которая возникает, если истекло время ожидания ответа. Это особенно важно при работе с функцией recvfrom, когда сервер или клиент ожидает данных, но не получает их в отведенное время.

SOCKET_ERROR — общий код ошибки для всех операций с сокетами. После его получения необходимо вызвать **WSAGetLastError()**, чтобы получить детализированный код ошибки.

Применение в работе:

Обработка ошибок позволяет программе устойчиво работать при сбоях в сети (например, если сообщение не было получено из-за проблем с подключением).

В случае **WSAETIMEDOUT**, клиент или сервер могут повторно отправить запрос или сообщение.

5. **Широковещательная передача:**

Широковещание (Broadcast) — это метод передачи данных, при котором одно сообщение отправляется всем узлам в локальной сети.

Особенности:

Адресатами широковещательных сообщений становятся все устройства в сети.

Требуется разрешить широковещательную передачу на сокете с помощью опции **SO_BROADCAST**, чтобы отправлять сообщения на широковещательный адрес.

Применение в работе:

Клиент отправляет запрос на широковещательный адрес сети (например, 192.168.1.255), чтобы найти доступные серверы.

6. **Многократное взаимодействие с клиентами:**

Для работы с несколькими клиентами сервер должен быть готов обрабатывать запросы многократно и одновременно. Это достигается за счет:

Циклического приема запросов — сервер постоянно ожидает новые запросы в бесконечном цикле, проверяя каждый запрос на соответствие заданному позывному.

Применение в работе:

Сервер выполняет функцию ожидания запросов от клиентов в бесконечном цикле. Как только клиент отправляет запрос с позывным, сервер отвечает. Цикл продолжается для обработки других клиентов.

7. **Структуры данных для хранения адресов:**

Для работы с сетевыми приложениями необходимо правильно использовать структуры данных для хранения IP-адресов и портов.

Основные структуры:

SOCKADDR_IN — структура, которая используется для хранения адреса IPv4. Содержит следующие поля:

- sin_family — тип адреса (для IPv4 это AF_INET).
- sin_port — номер порта (используется для назначения порта, на который сервер принимает запросы).
- sin_addr.s_addr — IP-адрес в виде 32-битного числа.

Применение в работе:

Клиент и сервер используют эту структуру для задания IP-адреса и порта, на которые отправляются и с которых принимаются данные.

8. **Конвертация данных:**

Для отображения IP-адресов и портов в удобном для человека виде используются следующие функции:

inet_ntoa() — преобразует IP-адрес из числового формата (как он хранится в sin_addr.s_addr) в строковый формат, который можно вывести на экран (например, 192.168.1.10).

ntohs() — преобразует номер порта из сетевого порядка байтов в порядок байтов, используемый на компьютере.

Применение в работе:

После получения ответа от сервера клиент выводит IP-адрес и номер порта сервера, преобразуя их в удобочитаемый формат.

9. **Клиент-серверное взаимодействие:**

Клиент отправляет запрос с позывным (например, "Hello") на сервер. Сервер получает запрос, проверяет правильность позывного и отправляет подтверждение обратно клиенту.

Применение в работе:

Клиент использует функцию **GetServer** для отправки широковещательного запроса.

Сервер использует функции **GetRequestFromClient** и **PutAnswerToClient** для приема запроса и отправки ответа клиенту.

5 лаба

Клиентская часть

1. Подключение библиотек:

```
#include "Winsock2.h"
#pragma comment(lib, "WS2_32.lib")
#include <ws2tcpip.h>
#include <chrono>
#include <iostream>
#include <string>
```

Библиотека `Winsock2.h` предоставляет функции для работы с сокетами.

Директива `#pragma comment(lib, "WS2_32.lib")` связывает библиотеку Winsock 2 при компиляции.

ws2tcpip.h необходим для работы с адресами и сетевыми функциями.

Остальные библиотеки используются для работы с вводом-выводом и строками.

2. Функция для получения текстового сообщения об ошибке:

```
string GetErrorMsgText(int code) {
    // Перечисление кодов ошибок и их описание
}
```

Функция предназначена для преобразования кодов ошибок в человеко-читаемые сообщения. Это помогает в отладке и понимании, что пошло не так в случае возникновения ошибок.

3. Функция для получения сервера по имени:

```
   bool GetServerByName(char* name, char* call, struct sockaddr* from, int*
flen) {
      SOCKET sS = socket(AF_INET, SOCK_DGRAM, NULL);  //
Создание сокета
      struct addrinfo hints, * res;
      ZeroMemory(&hints, sizeof(hints));
      hints.ai_family = AF_INET;
      hints.ai_socktype = SOCK_DGRAM;

      if (getaddrinfo(name, nullptr, &hints, &res) != 0) {
         throw SetErrorMsgText("getaddrinfo:", WSAGetLastError());
      }

      char ip_addr[INET_ADDRSTRLEN];
      inet_ntop(AF_INET, &((struct sockaddr_in*)res->ai_addr)->sin_addr,
ip_addr, INET_ADDRSTRLEN); // Получение IP-адреса сервера

      // Отправка сообщения на сервер
      int sent_len;
      if ((sent_len = sendto(sS, call, strlen(call), 0, (sockaddr*)&server_addr,
sizeof(server_addr))) == SOCKET_ERROR) {
         // Проверка на ошибки при отправке
      }

      // Установка таймаута для получения ответа
      int timeout = 20000; // 20 секунд
      if (setsockopt(sS, SOL_SOCKET, SO_RCVTIMEO, (char*)&timeout,
sizeof(timeout)) == SOCKET_ERROR) {
         // Проверка на ошибки
      }

      // Получение ответа от сервера
      int recv_len;
      char buffer[50];
      if ((recv_len = recvfrom(sS, buffer, sizeof(buffer) - 1, 0, from, flen)) ==
SOCKET_ERROR) {
         // Проверка на ошибки при получении
      }

      // Сравнение полученного ответа с отправленным сообщением
      if (strcmp(buffer, call) == 0) {
         closesocket(sS);
         return true;
```

```
    } else {
        closesocket(sS);
        return false;
    }
}
```

Создание сокета: Сокет создается для работы с протоколом UDP.

Получение IP-адреса: getaddrinfo используется для преобразования имени сервера в IP-адрес.

Отправка сообщения: sendto отправляет данные на сервер.

Таймаут: Установка таймаута для ожидания ответа предотвращает бесконечное ожидание.

Получение ответа: recvfrom ожидает ответа от сервера и помещает его в буфер.

Сравнение ответов: Если полученный ответ совпадает с отправленным сообщением, сокет закрывается, и функция возвращает true, иначе — false.

4. Функция main():

Инициализация Winsock.

Установка параметров сервера и вызов функции GetServerByName.

Печать IP-адреса и порта сервера или сообщение об ошибке.

Серверная часть

1. Подключение библиотек:

```
#include "Winsock2.h"
#pragma comment(lib, "WS2_32.lib")
#include <ws2tcpip.h>
#include <string>
#include <vector>
#include <iostream>
```

Подключаются те же библиотеки, что и в клиентской части, поскольку сервер также использует сокеты для сетевого взаимодействия.

2. Функция для получения текстового сообщения об ошибке:

Аналогична функции в клиенте, позволяет преобразовать коды ошибок в текст.

3. Основной цикл сервера:

Создание сокета.

Настройка адреса и порта для прослушивания.

Ожидание сообщений от клиентов.

При получении сообщения сервер отвечает клиенту.

Общее описание работы

1. Клиент:

Создает сокет для UDP и запрашивает IP-адрес сервера по его имени.

Отправляет сообщение серверу и ожидает ответ в течение 20 секунд.

Обрабатывает полученные данные, сравнивая их с отправленным сообщением.

2. Сервер:

Создает сокет и ждет сообщений от клиентов.

При получении сообщения отправляет ответ обратно клиенту, поддерживая взаимодействие.

Обработка ошибок: В коде присутствует хорошая обработка ошибок, что позволяет избежать сбоев в случае возникновения непредвиденных ситуаций.

Использование UDP: Программа использует протокол UDP, который не гарантирует доставку данных. Это может быть актуально для приложений, где важна скорость передачи.

Тайм-ауты: Установка таймаутов помогает избежать бесконечного ожидания ответа от сервера, улучшая отзывчивость приложения.

Этот код представляет собой простой пример клиент-серверного взаимодействия через UDP с использованием Winsock на C++.