Модуль 1: Введение в сети

Задачи для самостоятельной отработки (3 штуки)

Задача 1: “Анализатор URL” Напишите программу, которая принимает от пользователя URL (например, http://example.com:8080/path/to/page), разбирает его и выводит отдельно: * Протокол (если указан) * Доменное имя (хост) * Порт (если указан, иначе вывести “по умолчанию для протокола”) * Путь

Подсказка: Используйте класс Uri.

Задача 2: “Сканер портов (очень простой)” Создайте программу, которая проверяет доступность нескольких заданных портов (например, 80, 443, 21, 22) на указанном пользователе хосте (например, localhost). Программа должна поочередно пытаться установить TCP-соединение с каждым портом (как в Задании 5) и выводить результат: “Порт [номер] открыт” или “Порт [номер] закрыт”.

Подсказка: Оберните подключение в цикл и try-catch. Используйте таймаут, чтобы не ждать слишком долго.

Задача 3: “Сравнение времени ответа” Напишите программу, которая отправляет по 3 ping-запроса на два разных хоста (например, google.com и yahoo.com), вычисляет среднее время ответа для каждого и выводит, какой хост ответил в среднем быстрее.

Подсказка: Используйте Ping и цикл, сохраняйте результаты в список для последующего расчета.

Модуль 2: Сокеты

Задачи для самостоятельной отработки (3 штуки)

Задача 1: “Сервер с фиксированным ответом” Создайте TCP-сервер, который при любом входящем подключении немедленно отправляет клиенту фиксированное приветственное сообщение (например, “Добро пожаловать на сервер!”) и затем закрывает соединение. Клиент должен только подключиться, получить сообщение и отобразить его.

Подсказка: Серверу не нужно вызывать Receive, только Accept и сразу Send.

Задача 2: “Клиент для измерения задержки (Ping-клиент)” Напишите TCP-клиент, который подключается к серверу, отправляет один пакет и

замеряет время между отправкой и получением ответа (RTT - Round Trip Time). Для этого используйте Stopwatch из пространства имен System.Diagnostics.

Подсказка:

```
var stopwatch = Stopwatch.StartNew();
// ... операция Send ...
// ... операция Receive ...
stopwatch.Stop();
Console.WriteLine($"RTT: {stopwatch.ElapsedMilliseconds} мс");
```

Задача 3: “Сервер с логированием” Модифицируйте сервер из Задания 4 так, чтобы он не только обрабатывал клиентов, но и записывал в текстовый файл лог всех действий: время подключения клиента, его номер, полученное сообщение и время отключения.

Подсказка: Используйте StreamWriter и DateTime.Now для записи в файл.

Модуль 3: TCP и UDP сокеты

Задачи для самостоятельной отработки (3 штуки)

Задача 1: “Сервер времени” Создайте два сервера: TCP и UDP, которые возвращают текущее время сервера. TCP-сервер должен поддерживать multiple клиентов одновременно, а UDP-сервер должен обрабатывать запросы от разных клиентов. Клиент должен иметь возможность выбирать, к какому серверу подключаться.

Подсказка: Используйте DateTime.Now для получения текущего времени. Для TCP-сервера используйте многопоточность из Задания 5.

Задача 2: “Измеритель потерь пакетов” Напишите программу, которая отправляет 100 UDP-пакетов на сервер и подсчитывает процент потерянных пакетов. Программа должна выводить статистику: отправлено, получено, потеряно, процент потерь.

Подсказка: Модифицируйте Задание 3, добавив подсчет статистики. Сервер должен отвечать на каждый пакет.

Задача 3: “Простой чат” Реализуйте простой консольный чат на TCP, где сервер может отправлять сообщения всем подключенным клиентам. Клиенты

должны видеть сообщения от других клиентов. Сервер должен иметь отдельный поток для ввода сообщений от администратора.

Подсказка: Вам понадобится хранить список всех подключенных клиентских сокетов на сервере и рассылать сообщения всем клиентам при получении нового сообщения от любого из них или от администратора.

Модуль 4: Unicast, Broadcast, Multicast

Задачи для самостоятельной отработки (3 штуки)

Задача 1: “Сетевой discovery сервис” Создайте систему, где сервисы могут объявлять о своем существовании через broadcast, а клиенты могут их обнаруживать. Сервис должен каждые 5 секунд отправлять broadcast сообщение с информацией о себе (имя, тип, IP-адрес). Клиент должен собирать список доступных сервисов и выводить его.

Подсказка: Используйте JSON для сериализации информации о сервисе. Клиент должен обновлять список сервисов при получении broadcast сообщений.

Задача 2: “Multicast чат-комната” Реализуйте простой чат, где пользователи могут присоединяться к multicast группе и обмениваться сообщениями. Каждое сообщение должно содержать имя пользователя и временную метку. Реализуйте команды для смены имени пользователя и выхода из чата.

Подсказка: Используйте отдельные потоки для отправки и получения сообщений. Для отправки сообщений от пользователя используйте Console.ReadLine() в основном потоке.

Задача 3: “Измеритель сетевой нагрузки” Создайте программу, которая измеряет нагрузку на сеть при использовании разных типов рассылки. Программа должна отправлять определенное количество пакетов каждого типа (unicast, broadcast, multicast) и измерять: общее время отправки, среднее время на пакет, процент потерь (для multicast).

Подсказка: Запустите несколько экземпляров получателей для каждого типа рассылки. Используйте Stopwatch для точного измерения времени.

Модуль 5: HTTP, SMTP, FTP

Задачи для самостоятельной отработки (3 штуки)

Задача 1: “Веб-сканер заголовков” Создайте программу, которая принимает список URL-адресов и для каждого проверяет: - Статус код ответа - Наличие заголовков безопасности (X-Frame-Options, X-Content-Type-Options) - Версию HTTP-сервера - Поддержку HTTPS (редирект с HTTP на HTTPS)

Подсказка: Используйте HttpClient с отключенным автоматическим редиректом, чтобы анализировать исходные ответы. Обрабатывайте разные коды состояния (301, 302, 404, 500).

Задача 2: “Умный email-клиент” Напишите программу для отправки email с поддержкой: - Отправки нескольким получателям - HTML-формата писем - Вложений (файлов) - Обработки ошибок SMTP-сервера

Подсказка: Используйте классы MailMessage, Attachment. Для HTML-писем установите свойство IsBodyHtml = true и укажите альтернативное текстовое представление.

Задача 3: “FTP-менеджер” Создайте консольную утилиту для работы с FTP-сервером, поддерживающую команды: - list - показать содержимое текущей директории - download <filename> - скачать файл - upload <localfile> - загрузить файл на сервер - delete <filename> - удалить файл с сервера - cd <directory> - сменить директорию

Подсказка: Используйте разные методы WebRequestMethods.Ftp для каждой операции. Для навигации по директориям потребуется отслеживать текущий путь