Модуль 1: Введение в сети

Теоретическая часть

1. Основные понятия:

* **Сеть** — совокупность устройств (хостов), соединенных каналами связи для обмена данными.

* **IP-адрес** — уникальный сетевой адрес узла в сети, построенной по стеку TCP/IP. Пример: 192.168.1.10.

* **Порт** — числовой идентификатор (от 0 до 65535), определяющий конкретное приложение или процесс на узле. Пример: 80 для HTTP.

* **Локальный хост (localhost)** — стандартное имя для обращения к своему собственному компьютеру. Соответствует IP-адресу 127.0.0.1.

2. Модель Взаимодействия Открытых Систем (OSI) и TCP/IP:

* **OSI** — теоретическая 7-уровневая модель.

* **TCP/IP** — практический 4-уровневый набор сетевых протоколов, на котором построен Интернет.

* **Прикладной уровень (Application):** HTTP, FTP, SMTP. Отвечает за взаимодействие между приложениями.

* **Транспортный уровень (Transport):** TCP, UDP. Обеспечивает передачу данных между приложениями на разных хостах.

* **Сетевой уровень (Internet):** IP. Отвечает за маршрутизацию и доставку пакетов по IP-адресам.

* **Канальный уровень (Link):** Работа с сетевыми интерфейсами (Ethernet, Wi-Fi).

3. Ключевые протоколы транспортного уровня:

* **TCP (Transmission Control Protocol):**

    * *С установлением соединения* (handshake).

    * *Надежный*: гарантирует доставку и порядок пакетов.

    * *Потоковый*: данные передаются как непрерывный поток байтов.

* *Примеры*: веб-браузинг, электронная почта, передача файлов.

* **UDP (User Datagram Protocol):**

* *Без установления соединения*.

* *Ненадежный*: нет гарантий доставки и порядка.

* *Дейтаграммный*: данные передаются отдельными блоками (датаграммами).

* *Быстрый*.

* *Примеры*: видеостриминг, онлайн-игры, DNS-запросы.

4. Классы в .NET для сетевых операций: Пространство имен System.Net и System.Net.Sockets.

* IPAddress - представляет IP-адрес.

* IPEndPoint - представляет сетевую конечную точку (IP-адрес + порт).

* Dns - предоставляет методы для разрешения доменных имен в IP-адреса.

Модуль 2: Сокеты

Теоретическая часть

1. Что такое сокет?

* **Сокет** — это конечная точка для межпроцессного взаимодействия через сеть. Это абстракция, предоставляющая программисту универсальный интерфейс для отправки и получения данных по сети, скрывая сложность низкоуровневых протоколов.

* Можно представить сокет как “дверь” между приложением и сетью. Данные, отправляемые в сокет, уходят в сеть, а данные, приходящие из сети, забираются из сокета.

2. Класс System.Net.Sockets.Socket: Это основной класс в .NET для низкоуровневой работы с сетью. Он предоставляет полный контроль над сетевыми операциями.

* **Конструктор:** Для создания сокета указать:

* AddressFamily: Семейство адресов (например, InterNetwork для IPv4, InterNetworkV6 для IPv6).

* SocketType: Тип сокета (например, Stream для TCP, Dgram для UDP).

* ProtocolType: Тип протокола (например, Tcp для TCP, Udp для UDP).

3. Основные методы сокета:

* **Для сервера (прослушивающий сокет):**

* Bind(endPoint): Связывает сокет с локальной конечной точкой (IP-адрес и порт).

* Listen(backlog): Переводит сокет в состояние прослушивания входящих подключений. backlog — максимальная длина очереди ожидающих подключений.

* Accept(): Синхронно принимает входящее подключение. Возвращает *новый* сокет для обмена данными с подключившимся клиентом. Блокирует поток до появления подключения.

* AcceptAsync(): Асинхронная версия Accept.

* **Для клиента:**

* Connect(endPoint): Синхронно устанавливает соединение с удаленным хостом.

* ConnectAsync(): Асинхронная версия Connect.

* **Для обмена данными (и клиент, и сервер):**

* Send(buffer) / Receive(buffer): Синхронная отправка и получение данных через сокет.

* SendAsync() / ReceiveAsync(): Асинхронные версии. Рекомендуются для избежания блокировок потоков.

* **Общее:**

* Shutdown(SocketShutdown): Отключает отправку и/или получение данных.

* Close() / Dispose(): Закрывает сокет и освобождает ресурсы.

4. Важность асинхронных операций: Сетевые операции (ожидание подключения, отправка, получение) по своей природе медленные.

Использование синхронных методов (Accept, Receive) блокирует текущий поток, что неэффективно для масштабируемых серверов. Асинхронные методы (AcceptAsync, ReceiveAsync) освобождают поток для обработки других задач, пока ожидается сетевая операция.

Модуль 3: TCP и UDP сокеты

Теоретическая часть

1. Сравнение TCP и UDP:

Характеристика	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Соединение	Устанавливает соединение (handshake)	Без соединения
Надежность	Гарантированная доставка, контроль целостности, повторная отправка потерянных пакетов	Нет гарантий доставки, возможна потеря пакетов
Порядок	Сохраняет порядок пакетов	Порядок не гарантируется
Скорость	Медленнее из-за накладных расходов	Быстрее, меньше накладных расходов
Контроль перегрузки	Есть	Нет
Использование	Веб-браузинг, email, передача файлов	Видеостриминг, онлайн-игры, VoIP, DNS

2. TCP-сокеты (SocketType.Stream):

- Работают с потоками байтов
- Требуют предварительного установления соединения
- Используют буферизацию
- Идеальны для сценариев, где важна надежность

3. UDP-сокеты (SocketType.Dgram):

- Работают с отдельными датаграммами (пакетами)

- Не требуют соединения

- Каждая отправка/получение - независимая операция

- Идеальны для сценариев, где важна скорость

4. Ключевые методы для UDP:

- SendTo() / ReceiveFrom() - для работы с конкретными адресами

- Send() / Receive() - после вызова Connect()

Модуль 4: Unicast, Broadcast, Multicast

Теоретическая часть

1. Типы адресации в сетях:

- **Unicast (Одноадресная рассылка)**
  - Отправка данных от одного узла к одному узлу
  - Самый распространенный тип коммуникации
  - Пример: обычный HTTP-запрос к веб-серверу
  - Использует обычные IP-адреса: 192.168.1.10 → 192.168.1.20
- **Broadcast (Широковещательная рассылка)**
  - Отправка данных от одного узла ко всем узлам в сети
  - Ограничивается локальной сетью (не маршрутизируется)
  - Пример: ARP-запросы, DHCP-обнаружение
  - Специальные адреса:
    - 255.255.255.255 - ограниченный broadcast
    - 192.168.1.255 - directed broadcast для сети 192.168.1.0/24
- **Multicast (Групповая рассылка)**
  - Отправка данных от одного узла к группе узлов
  - Эффективнее, чем multiple unicast
  - Маршрутизируется (в отличие от broadcast)
  - Пример: видеостриминг, онлайн-конференции
  - Диапазон адресов: 224.0.0.0 - 239.255.255.255

2. Особенности реализации в .NET:

- **Broadcast:** Установка свойства Socket.EnableBroadcast = true

- **Multicast:** Использование классов UdpClient с методами JoinMulticastGroup()
- **Multicast-адреса:**
  - 224.0.0.0 - 224.0.0.255 - локальные (не маршрутизируются)
  - 224.0.1.0 - 238.255.255.255 - глобальные
  - 239.0.0.0 - 239.255.255.255 - административно ограниченные

3. TTL (Time To Live) для multicast:

- Определяет, через сколько маршрутизаторов может пройти пакет
- Значение по умолчанию: 1 (только локальная сеть)
- Устанавливается через Socket.SetSocketOption()

Модуль 5: HTTP, SMTP, FTP

Теоретическая часть

1. Прикладные протоколы уровня Application:

- **HTTP (HyperText Transfer Protocol)**
  - Протокол для передачи веб-страниц и API-запросов
  - Порт по умолчанию: 80 (HTTP), 443 (HTTPS)
  - Методы: GET, POST, PUT, DELETE и др.
  - Статус-коды: 200 (OK), 404 (Not Found), 500 (Server Error)
- **SMTP (Simple Mail Transfer Protocol)**
  - Протокол для отправки электронной почты
  - Порт по умолчанию: 25, 587 (TLS)
  - Команды: HELO, MAIL FROM, RCPT TO, DATA, QUIT
- **FTP (File Transfer Protocol)**
  - Протокол для передачи файлов
  - Использует два соединения: управляющее (порт 21) и данных (порт 20)
  - Команды: USER, PASS, LIST, RETR, STOR

2. Классы .NET для работы с прикладными протоколами:

- **Для HTTP:**
  - HttpClient - современный класс для HTTP-запросов

    - HttpWebRequest / HttpWebResponse - legacy классы
    - WebClient - упрощенный клиент
- **Для SMTP:**
    - SmtpClient - для отправки email
    - Пространство имен: System.Net.Mail
- **Для FTP:**
    - FtpWebRequest / FtpWebResponse
    - WebClient с указанием протокола

3. Основные концепции:

- **HTTP-запросы:** Заголовки, тело, методы
- **SMTP-сессии:** Диалог с почтовым сервером
- **FTP-сессии:** Разделение управляющего и data-каналов

Заключение

Поздравляю! Вы прошли полный курс по сетевому программированию в .NET. Вы изучили:

1. **Основы сетей** - IP-адреса, порты, TCP/UDP
2. **Работу с сокетами** - низкоуровневое сетевое программирование
3. **TCP и UDP протоколы** - их различия и применение
4. **Типы сетевой рассылки** - Unicast, Broadcast, Multicast
5. **Прикладные протоколы** - HTTP, SMTP, FTP

Каждый модуль содержал как практические задания с готовыми решениями, так и задачи для самостоятельной работы. Это дает прочную основу для дальнейшего развития в области сетевого программирования.

Для дальнейшего изучения рекомендую обратить внимание на:

- ASP.NET Core для создания веб-API

- SignalR для real-time коммуникации

- gRPC для высокопроизводительных RPC-сервисов

- WebSockets для двусторонней связи

Удачи в дальнейшем обучении!