Принципы работы в сети

Сетевая модель

- Для единого представления данных в сетях с неоднородными устройствами и программным обеспечением международная организация по стандартам ISO (International Standardization Organization) разработала базовую модель связи открытых систем OSI (Open System Interconnection)

Сетевая модель OSI

Сетевое приложение

- Вся сеть состоит из отдельных элементов - хостов, которые представляют собой компьютеры и другие подключенные устройства. Между собой они соединены каналами связи (кабели Ethernet, Wi-Fi и т.д.) и маршрутизаторами. Маршрутизаторы объединяют компьютеры в подсети и контролируют передачу данных между ними.

Сетевое приложение

- Для взаимодействия компьютеры применяют **протоколы**.
- Протокол представляет собой соглашения о том, как пакеты данных будут передаваться по каналам коммуникации.

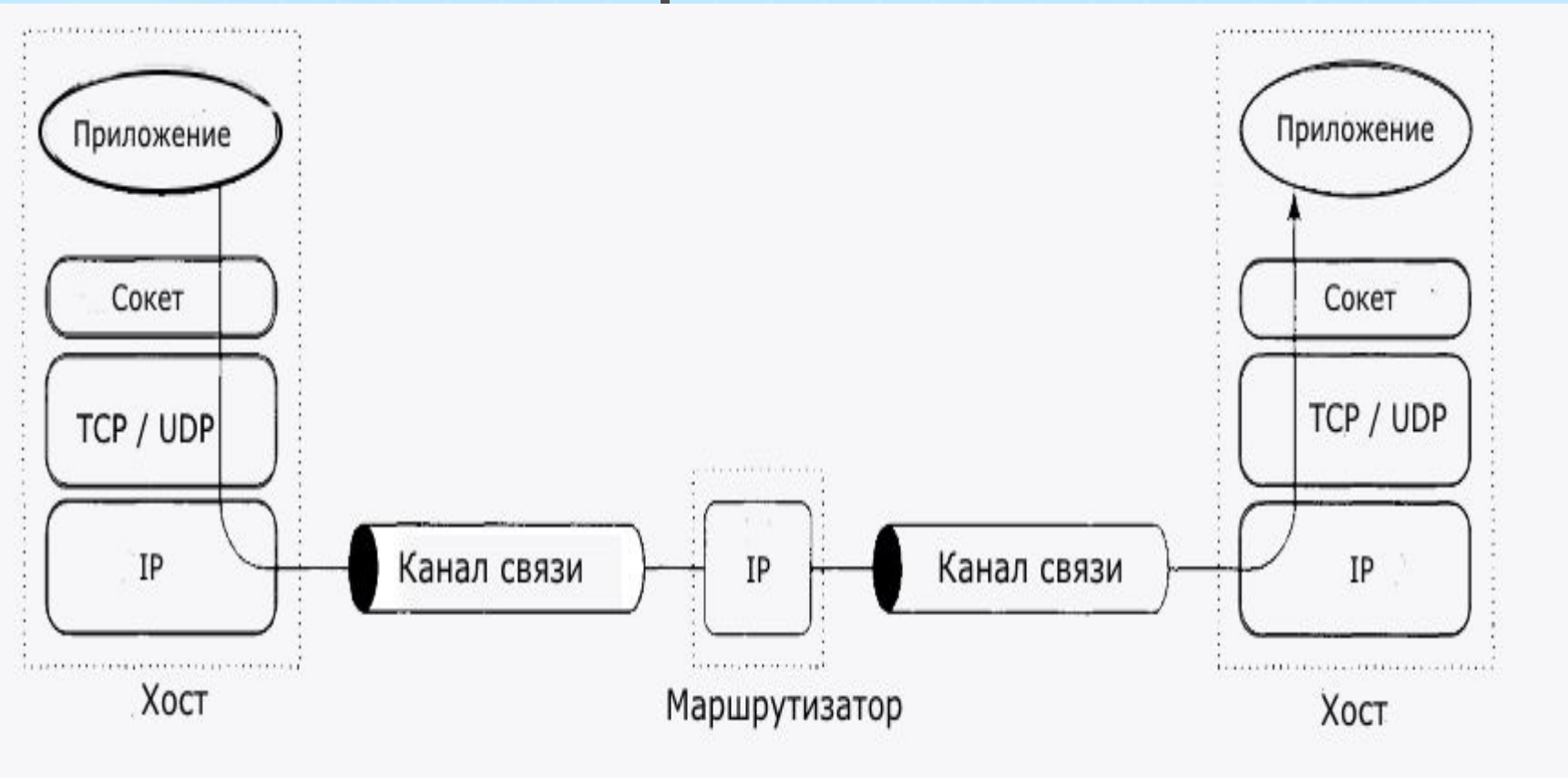
Сетевое приложение

- Существует множество различных протоколов. Протоколы, которые используются для передачи данных по сети, составляют семейство протоколов TCP/IP. Основные из них: Internet Protocol (IP), Transmission Control Protocol (TCP) и User Datagram Protocol (UDP).

Сетевое приложение

- Существует множество различных протоколов. Протоколы, которые используются для передачи данных по сети, составляют семейство протоколов TCP/IP. Основные из них: Internet Protocol (IP), Transmission Control Protocol (TCP) и User Datagram Protocol (UDP).

Условная схема работы сетевого приложения



Сетевое приложение

- IP представляет сетевой уровень. Он использует нижележащие уровни, которые представляют физические каналы коммуникации - кабели Ethernet и т.д., для передачи пакетов с данными другому хосту.
- Выше IP располагается транспортный уровень, который образуют протоколы TCP и UDP.

Протоколы

- Протоколы используют определенные порты для передачи данных.
- TCP позволяет отследить потерю пакетов и их дублирование при передаче.
- UDP подобного не позволяет сделать и нацелен на простую передачу данных.

Сокеты

- *Приложение взаимодействует с уровнем TCP / UDP не напрямую, а через специальный API, который предоставляют* ***сокеты****.*
- Т.е. сокеты – это интерфейс для создания сетевых приложений, который опирается на встроенные возможности операционной системы.

Виды сокетов

- потоковые сокеты (stream socket)
- дейтаграммные сокеты (datagram socket).
- Потоковые сокеты используют протокол TCP, дейтаграммные - протокол UDP.

Службы DNS

- выполняют сопоставление между интернет-адресами в формате IPv4 или IPv6 и доменными названиями.
- Пример:

"www.microsoft.com", ему соответствует адрес в формате IPv4 2.23.143.150

Порт

- Кроме адреса при сетевых взаимодействиях используются **порты**. Порт представляет 16-битное число в диапазоне от 1 до 65 535. Использование портов позволяет разграничить несколько запущенных приложений на одном хосте.

IPAddres свойсва и методы

- Метод Parse(): преобразует строковое представление адреса в IPAddress

IPAddress ip = IPAddress.Parse("127.0.0.1"); // ip указывает на локальный адрес

- Статическое свойство Loopback: возвращает объект IPAddress для адреса 127.0.0.1. Аналогично вышеприведенному коду
- Статическое свойство Any: возвращает объект IPAddress для адреса 0.0.0.0
- Статическое свойство Broadcast: возвращает объект IPAddress для адреса 255.255.255.255

IPHostEntry

- Также для получения адреса в сети используется класс **IPHostEntry**. Он содержит информацию об определенном компьютере-хосте.
- С помощью свойства **HostName** этот класс возвращает имя хоста, а с помощью свойства **AddressList** - все ip-адреса хоста, так как один компьютер может иметь в сети несколько ip-адресов.
- Для взаимодействия с dns-сервером и получения ip-адреса применяется класс **Dns**. Для получения информации о хосте компьютера и его адресах у него имеется метод **GetHostEntry()**

Пример

```
IPHostEntry host1 = Dns.GetHostEntry("www.microsoft.com");
Console.WriteLine(host1.HostName);
foreach (IPAddress ip in host1.AddressList)
    Console.WriteLine(ip.ToString());

Console.WriteLine();

IPHostEntry host2 = Dns.GetHostEntry("google.com");
Console.WriteLine(host2.HostName);
foreach (IPAddress ip in host2.AddressList)
    Console.WriteLine(ip.ToString());
```

```
e10088.dspb.akamaiedge.net
2.23.143.150

google.com
178.35.137.101
178.35.137.106
178.35.137.80
178.35.137.88
178.35.137.102
178.35.137.123
178.35.137.117
178.35.137.110
178.35.137.95
178.35.137.113
178.35.137.84
178.35.137.99
178.35.137.112
178.35.137.121
178.35.137.90
178.35.137.91
```

Загрузка файлов WebClient

- Самый простой способ загрузки предоставляет метод **DownloadFile()**. Например, загрузим файл с какого-нибудь сайта:
- WebClient client = new WebClient();
- client.DownloadFile("http://somesite.com/book.pdf", "myBook.pdf");
- Console.WriteLine("Файл загружен");

Загрузка файлов WebClient

```
WebClient client = new WebClient();

using (Stream stream = client.OpenRead("http://somesite.com/sometext.txt"))
{
    using (StreamReader reader = new StreamReader(stream))
    {
        string line = "";
        while ((line = reader.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
}

Console.WriteLine("Файл загружен");
Console.Read();
```

Отправка запросов **WebRequest** и **WebResponse**

Класс **WebRequest** служит для отправки запроса.

Класс **WebResponse** служит для получения ответа.

Отправка запросов **WebRequest** и **WebResponse**

Принцип работы сводится к

1. Создание объекта WebRequest с помощью метода Create(), в который передается адрес ресурса с виде строки или объекта Uri
2. Отправка запроса и получение ответа
3. Получение потока ответа и манипуляции с ним

Отправка запросов

```
WebRequest request = WebRequest.Create("http://somesite.com/myfile.txt");
WebResponse response = request.GetResponse();
using (Stream stream = response.GetResponseStream())
{
    using (StreamReader reader = new StreamReader(stream))
    {
        string line = "";
        while ((line = reader.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
}
response.Close();
Console.WriteLine("Запрос выполнен");
Console.Read();
```