Информатика

Network

Вспомним…

- Интернет и Web (WWW)
  - Инфраструктура и сеть гипертекстовых документов
- Сеть – хосты, соединенные каналами связи и объединенные в подсети маршрутизаторами
- Общение хостов по сетевым протоколам
- Протоколы организованы в уровневую модель

Модель OSI

- Open Systems Interconnection basic reference model
  - Эталонная Модель Взаимодействия Открытых Систем
- 7 уровней взаимодействия компьютеров в сетях
  - Со своими форматами данных
  - Со своими правилами передачи (**протоколами**)

Модель OSI

	Уровень (layer)	Тип данных (PDU)	Функции	Примеры
Host layers	7. Прикладной (application)		Доступ к сетевым службам	HTTP, FTP, SMTP, RDP, SNMP
	6. Представления (presentation)	Данные	Представление и шифрование данных	ASCII, EBCDIC, JPEG
	5. Сеансовый (session)		Управление сеансом связи	RPC, PAP
	4. Транспортный (transport)	Сегменты (segment)/ Дейтаграммы (datagram)	Прямая связь между конечными пунктами и надёжность	TCP, UDP, SCTP, PORTS
	3. Сетевой (network)	Пакеты (packet)	Определение маршрута и логическая адресация	IPv4, IPv6, IPsec, AppleTalk
Media layers	2. Канальный (data link)	Биты (bit)/ Кадры (frame)	Физическая адресация	PPP, IEEE 802.22, Ethernet, DSL, ARP, L2TP, Network Cards
	1. Физический (physical)	Биты (bit)	Работа со средой передачи, сигналами и двоичными данными	USB, витая пара, коаксиальный кабель, оптический кабель

Сетевой уровень и выше

- В основном работаем с сетевым, транспортным и прикладным уровнями
- IP – маршрутизация пакетов
- TCP – надежная передача данных по маршруту
- UDP – быстрая передача данных по маршруту без каких-либо гарантий
- HTTP – протокол приложений поверх TCP

HTTP сервер

- Мы рассматривали HTTP-серверы
- В основе – HttpListener (System.Net)
- HttpListener
  - Работает с контекстом (Context)
  - Принимает запрос (Request)
  - Отправляет ответ (Response)

Клиенты http-сервера

- Мы работали с клиентами – html-страницами
- В запросах – данные формы
  - Или работа AJAX
- А что, если нужно из приложения?

HTTP клиенты

- .NET Framework предлагает несколько стандартных классов, отличающихся уровнем абстракции
  - HttpWebRequest
  - WebClient
  - HttpClient
- Также есть много сторонних
  - RestSharp, например

HttpWebRequest

- Самый старый из троицы
- Даёт ***полный*** контроль над объектами запроса/ответа
  - Заголовками, таймаутами, куки…
- Работает с отдельным потоком

Сложность

- Богатые возможности и полный контроль приводят к сложности работы с HttpWebRequest
- Это может привести к ошибкам во множестве сценариев
- Лучше не работать с ним, если вам не нужны эти низкоуровневые возможности и полный контроль

Использование

- Для отправки запроса и приёма ответа используются классы WebRequest и WebResponse
  - Это-базовые классы для HttpWebRequest и HttpWebResponse соответственно
- Также WebRequest – фабрика HttpWebRequest’ов

Пример

```
var request =
WebRequest.CreateHttp("https://reqres.in/api/users/2");
var response = await request.GetResponseAsync();
using (var stream = response.GetResponseStream())
using (var reader = new StreamReader(stream))
{
    var userJson = await reader.ReadToEndAsync();
    Console.WriteLine(userJson);
}
response.Close();
```

Базовая HTTP-аутентификация

- user:password@host:port
- NetworkCredential – логин, пароль

```
HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://host.com/");
request.Credentials = new NetworkCredential(“login", “pass");
HttpWebResponse response = (HttpWebResponse)await request.GetResponseAsync();
```

WebClient

- Простой инструмент
  – более высокого уровня абстракции
- Надстройка над HttpWebRequest
- Упрощает решение большинства типовых задач
- Требует меньше кода

WebClient

- Доступен с .NET Core 2.0
- Появилась поддержка асинхронных методов (раньше не было)
- Есть отчеты загрузок

Пример: загрузка файла

```
var link =
    "https://media-www-asp.azureedge.net/
     media/5245130/home-hero-2.png";
var bytes = client.DownloadData(link);
var name = System.IO.Path.GetFileName(link);
System.IO.File.WriteAllBytes(name, bytes);
```

Пример: заголовки

```
client.Encoding=Encoding.UTF8;
client.QueryString.Add("q", "search text");
client.Headers.Add("User-Agent", "Mozilla/5.
                    Gecko/20101026 Firefox/3.6.12");
client.Headers.Add("Accept-Language", "ru");
var bytes =
    client.DownloadData("https://www.google.ru/search");
var page = Encoding.UTF8.GetString(bytes);
File.WriteAllText("search.html", page);
```

Пример: GET JSON

```
client.Encoding = Encoding.UTF8;

client.Headers.Add(HttpRequestHeader.ContentType,
                   "application/json");
client.QueryString.Add("page", "2");
var jsonData =
    client.DownloadString("https://reqres.in/api/users");
Console.WriteLine(jsonData);
Console.WriteLine();
```

Пример: POST JSON

```
client.Headers.Add(HttpRequestHeader.ContentType,
                   "application/json");
var json = new {email="peter@klaven",password="cityslicka"};
var str=Newtonsoft.Json.JsonConvert.SerializeObject(json);
var response =
    client.UploadString("https://reqres.in/api/login",str);
var tokenObj = JsonConvert.DeserializeAnonymousType(response,
                                                    new{token=""});
Console.WriteLine(tokenObj);
```

HttpClient

- Лучшее из «двух миров»
- Функциональность + чистота + многопоточность
- Один клиент – много запросов
- Лучше подходит для тестирования

Пример: POST JSON

```
using (var client = new HttpClient())
{
    var json = new { email = "peter@klaven", password = "cityslicka" };
    var str = Newtonsoft.Json.JsonConvert.SerializeObject(json);

    var content = new StringContent(str);
    content.Headers.ContentType =
        new MediaTypeHeaderValue("application/json");
    var result = await client.PostAsync("https://reqres.in/api/login", content);

    var response = await result.Content.ReadAsStringAsync();
    var tokenObj = JsonConvert.DeserializeAnonymousType(response,
                                                        new { token = "" });
    Console.WriteLine(tokenObj);
}
```

Помимо HTTP

- На пару уровней ниже
  - TCP
  - UDP
- Взаимодействуем по протоколам TCP и UDP с помощью сокетов (System.Net.Sockets.Socket)

Вспомогательные классы

System.Net.Dns и System.Net.IPAddress

```
var hostEntry = System.Net.Dns.GetHostEntry("microsoft.com");
Console.WriteLine(hostEntry.HostName);
foreach (System.Net.IPAddress ipAddress in hostEntry.AddressList)
    Console.WriteLine(ipAddress);


var hostByIp = Dns.GetHostEntry(IPAddress.Parse("127.0.0.1"));
Console.WriteLine(hostByIp.HostName);
foreach (var ipAddress in hostByIp.AddressList)
    Console.WriteLine(ipAddress);
Console.WriteLine();
```

Socket

- Предоставляет низкоуровневый интерфейс для приема/передачи данных по сети
- Содержит
  - AddressFamily – адреса сокета
  - ProtocolType – протокол сокета
  - SocketType – тип сокета (Dgram, Stream)
  - LocalEndPoint – адрес, по которой сокет принимает значения
  - Connected – подключен ли к удалённому хосту
  - RemoteEndPoint – адрес хоста, к которому подключен сокет

Примеры сокетов TCP, UDP

Мы можем комбинировать протоколы, типы сокета, семейства адресов

```
Socket tcpSocket =
new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);


Socket udpSocket =
new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
```

Основные методы

- **Accept**
  создаёт сокет
- **Bind**
  связывает с конечной точкой
- **Close**
  закрывает сокет
- **Connect**
  соединяется с удаленным хостом
- **Listen**
  начинает прослушку запросов
- **Poll**
  определяет состояние
- **Receive**
  получает данные
- **Send**
  отправляет данные
- **Shutdown**
  блокирует прием/отправку

TCP Socket

[Статья с примерами использования на Metanit]

Клиент и сервер

```
IPEndPoint point = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8000);

Socket listener =
    new Socket(AddressFamily.InterNetwork, SocketType.Stream,
    ProtocolType.Tcp);

listener.Bind(point);
listener.Listen(10);

while (true)
{
    Socket handler = listenSocket.Accept();
    StringBuilder builder = new StringBuilder();
    byte[] data = new byte[256]; // буфер для получаемых данных

    do
    {
        var bytes = handler.Receive(data);
        builder.Append(Encoding.Unicode.GetString(data, 0, bytes));
    } while (handler.Available > 0);

    Console.WriteLine(DateTime.Now.ToShortTimeString() + ": " + builder);

    string message = "message received";
    data = Encoding.Unicode.GetBytes(message);
    handler.Send(data);
    handler.Shutdown(SocketShutdown.Both);
    handler.Close();
}
```

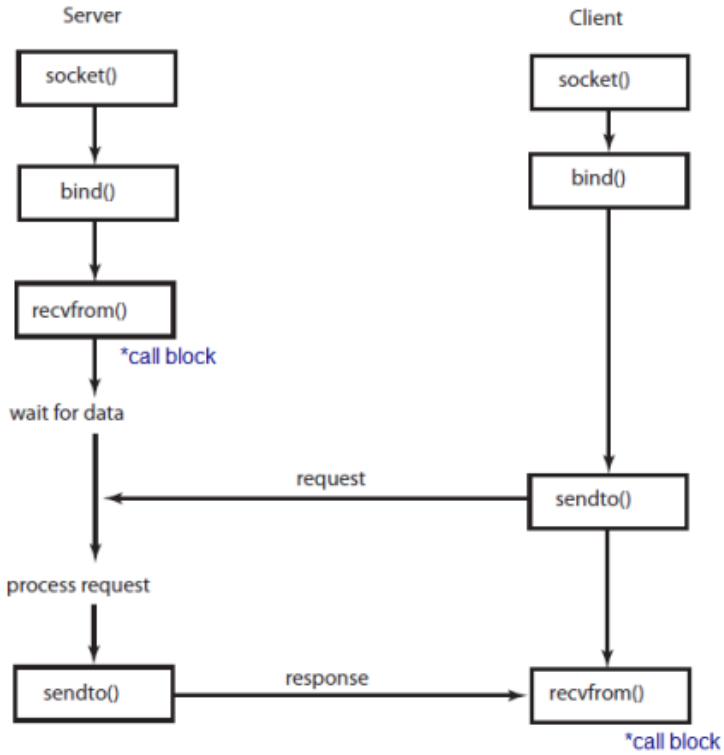
```
socket.Connect(poimt);
Console.Write("Введите сообщение:");
string message = Console.ReadLine();
byte[] data = Encoding.Unicode.GetBytes(message);
socket.Send(data);

// получаем ответ
data = new byte[256]; // буфер для ответа
StringBuilder builder = new StringBuilder();
int bytes = 0;
do
{
   bytes = socket.Receive(data, data.Length, 0);
    builder.Append(
        Encoding.Unicode.GetString(data, 0, bytes));
}
while (socket.Available > 0);
Console.WriteLine("ответ сервера: "  builder.ToString());

// закрываем сокет
socket.Shutdown(SocketShutdown.Both);
socket.Close();
```

UDP Socket

- После Bind не нужно вызывать Listen
- Приём и передача – **ReceiveFrom** и **SendTo**



[Статья с примером использования на Metanit]

Использование UDP сокетов

```
Socket socket = new Socket(AddressFamily.InterNetwork,
                           SocketType.Dgram, ProtocolType.Udp);
```

```
string message = Console.ReadLine();
byte[] data = Encoding.Unicode.GetBytes(message);
EndPoint remotePoint =
    new IPEndPoint(IPAddress.Parse("127.0.0.1"), port);
socket.SendTo(data, remotePoint);
```

```
byte[] data = new byte[256];
EndPoint remoteIp = new IPEndPoint(IPAddress.Any, 0);
int bytes = socket.ReceiveFrom(data, ref remoteIp);
```

Попроще сокетов

- Есть стандартные классы – надстройки над System.Net.Sockets.Socket
- Упрощают работу с протоколами TCP и UDP
- **TcpListener** и **TcpClient**
  - как HttpListener и HttpClient
- **UdpClient**

TcpListener

```
TcpListener server = null;

IPAddress localAddr = IPAddress.Parse("127.0.0.1");
server = new TcpListener(localAddr, 8080);

server.Start();

while (true)
{

    TcpClient client = server.AcceptTcpClient();
    NetworkStream stream = client.GetStream();

    string response = "Hello world!";
    byte[] data = Encoding.UTF8.GetBytes(response);

    stream.Write(data, 0, data.Length);
    client.Close();
}

server.Stop();
```

TcpClient

```
TcpClient client = new TcpClient();
client.Connect("127.0.0.1", 8080);

byte[] data = new byte[256];
StringBuilder response = new StringBuilder();
NetworkStream stream = client.GetStream();

do
{
    int size = stream.Read(data, 0, data.Length);
    response.Append(Encoding.UTF8.GetString(data));
}
while (stream.DataAvailable);

Console.WriteLine(response.ToString());

stream.Close();
client.Close();
```

Чат на TCP(Listener + Client)

- На сервере – список клиентов
  - Клиент – объект с полем потока
  - Регистрация клиента по AcceptTcpClient
- Сервер прослушивает клиентов, рассылает сообщения остальным
  - Читает/записывает поток

Пример кода

HttpListener и TcpListener

- HttpListener – высокоуровневое расширение TcpListener
- Чем дополняет?
- В чём особенности?

Упрощённый UDP

- Протокол UDP позволяет отправить данные (datagram) на удалённый узел
- Не нужен сервер, данные передаются напрямую между узлами
- Можно вещать данные множеству адресов в подсети

Приём и передача

- Методы Send и Receive

```
UdpClient client = new UdpClient();
string message = "Hello world!";
byte[] data = Encoding.UTF8.GetBytes(message);
client.Send(data, data.Length, "127.0.0.1", 8001);
client.Close();
```

```
UdpClient client = new UdpClient(8001);
IPEndPoint ip = null;
byte[] data = client.Receive(ref ip);
string message = Encoding.UTF8.GetString(data);
Console.WriteLine(message);
client.Close();
```

Вещание (multicast)

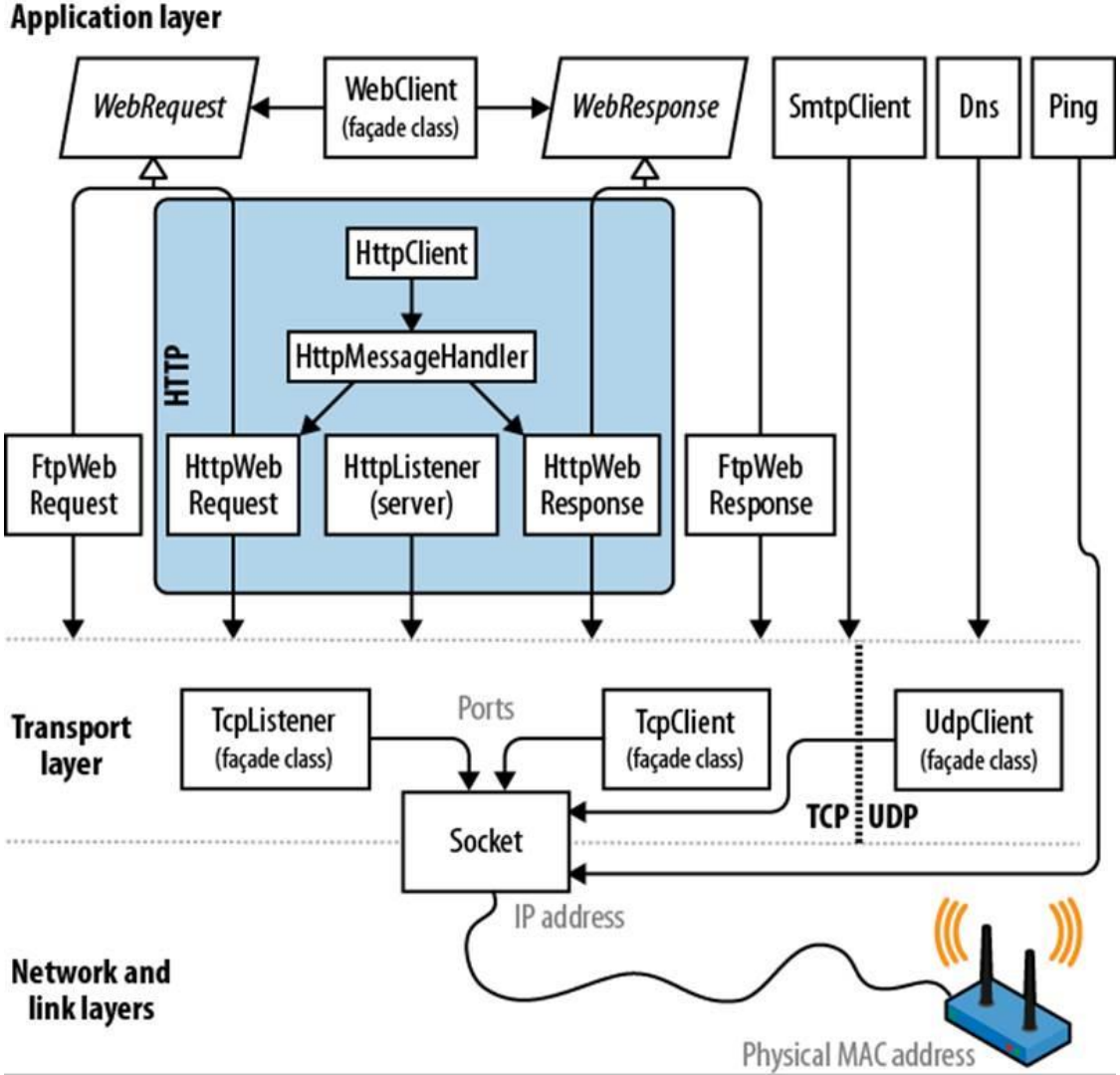
- Протокол UDP позволяет рассылать одно сообщение группе клиентов-получателей
- В отличие от TCP, не нужно рассылать сообщение каждому клиенту по отдельности
- Передача идёт не дальше локальных сетей (ограничивается маршрутизаторами)

Вещание

- Для работы вещания клиентам нужно присоединиться к группе, которой предназначается рассылка
- JoinMulticastGroup(адрес, хопы)

```
udpClient
    .JoinMulticastGroup(IPAddress.Parse("234.0.0.0"), 50);
```

Пример кода


Application layer
WebRequest
WebClient
(façade class)
WebResponse
SmtpClient
Dns
Ping
HttpClient
HttpMessageHandler
HTTP
FtpWeb
Request
HttpWeb
Request
HttpListener
(server)
HttpWeb
Response
FtpWeb
Response
Transport
layer
TcpListener
(façade class)
Ports
TcpClient
(façade class)
UdpClient
(façade class)
TCP
UDP
Socket
IP address
Network and
link layers
Physical MAC address

Почитать

- https://metanit.com/sharp/net/
- https://habrahabr.ru/post/209144/
- Albahari, C# 7 in a nutshell, Chapter 16

Вопросы?

e-mail: marchenko@it.kfu.ru