Networking

Ing. Michal Radecký, Ph.D.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

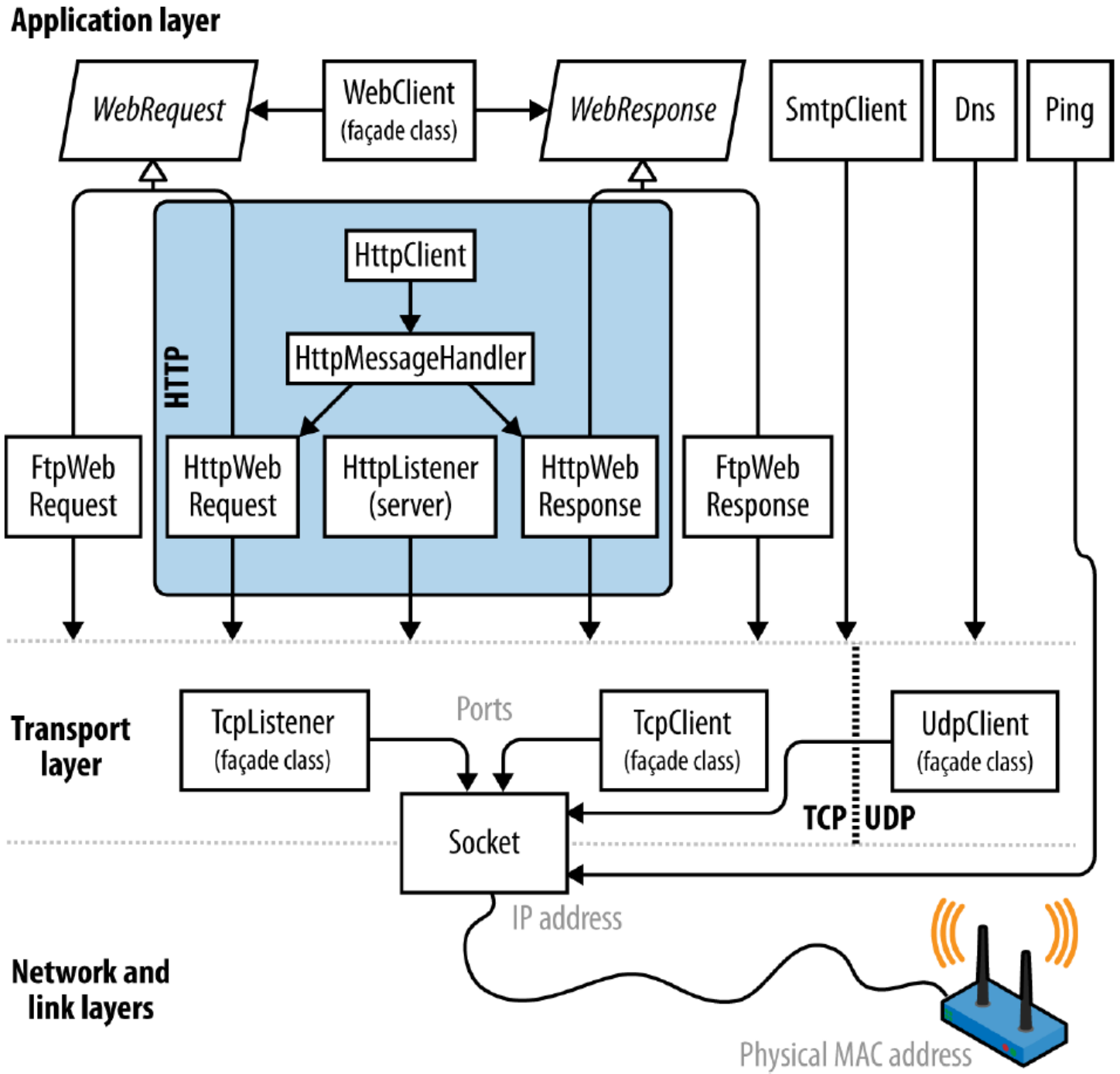
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Síťová komunikace

- Možnost implementace síťové komunikace napříč „.NETem“
- Namespace zajišťuje široké možnosti síťové komunikace na různých úrovních a z různých pohledů
- `WebClient` – fasádní třída (abstrakce) pro obecnou práci s HTTP/FTP protokolem
- `WebRequest`, `WebResponse` – implementace klientské části HTTP/FTP na nižší úrovni
- `HttpClient` – konzumace HTTP komunikace (API)
- `HttpListener` – třída pro implementaci HTTP serveru
- `SmtpClient` – třída pro tvorbu a zasílání emailových zpráv na SMTP protokolu
- `Dns` – třída pro využívání DNS systému
- `TcpClient`, `UdpClient`, `TcpListener`, `Socket` – další třídy pro přístup ke službám nižších vrstev síťové architektury
- CLR standardně nepodporuje konkurenci HTTP připojení (více než 1 požadavek najednou), je možné ale upravit nastavením `ServicePointManager.DefaultConnectionLimit`

- **`System.Net`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net?view=net-5.0

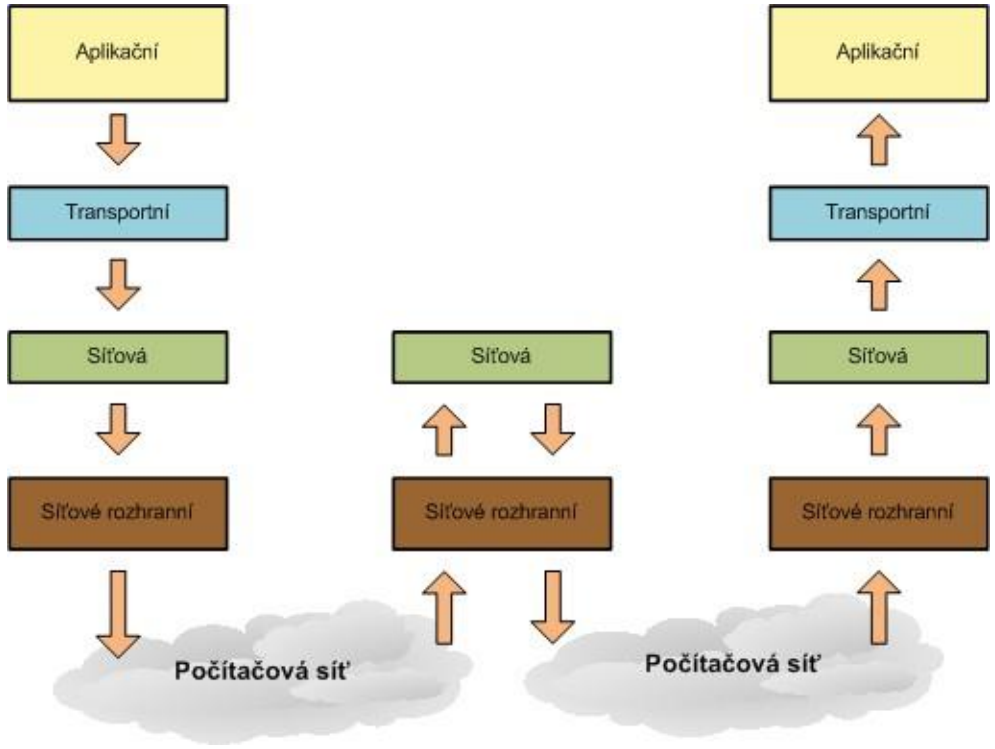
Ukázkové příklady na
http://www.albahari.com/nutshell/E9-CH16.aspx

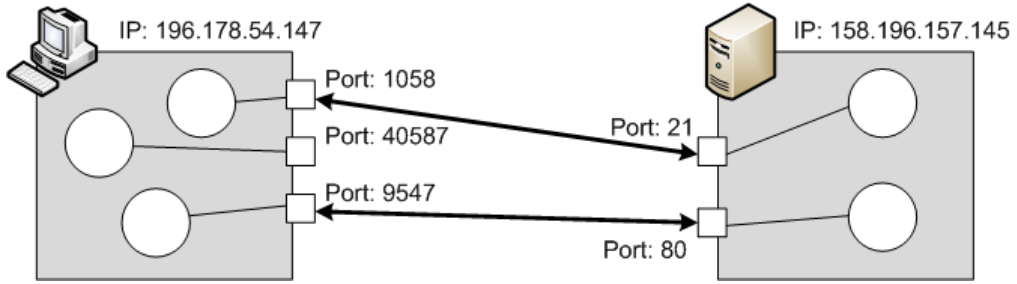


Zdroj: C# 7.0 in a Nutshell

Důležité pojmy

- IP adresa
  - Identifikace uzlu v počítačové síti využívající technologie IP (TCP/IP, UDP/IP)
  - Rozlišuje se IPv4 a IPv6
  - Odesilatel i příjemce
  - Transportní vrstva
  - `System.Net.IPAddress`
- Port
  - Identifikace aplikace v rámci uzlu/počítače
  - Na jednom počítači může posílat/přijímat komunikaci relativně neomezené množství různých aplikací
  - Určité porty jsou „vyhrazené“, např. 80, 21
  - `System.Net.IPEndPoint`
- URI (Unified Resource Identifier)
  - Řetězec jednoznačně identifikující abstraktní či fyzický zdroj
  - V kontextu komunikace se jedná o URL





URI

- Využívá se třída, která je schopna pracovat s URI různého rozsahu a poskytovat její části
- Často je objekt třídy URI vstupním parametrem celé řady metod (namísto řetězce jako takového)
- Vytvoření objektu s vlastnostmi na základě volání konstruktoru a vstupního řetězce
- Pro modifikaci vlastností nutné použít `UriBuilder`

```
Uri uri = new Uri("https://user:password@www.contoso.com:80/Home/Index.htm?q1=v1&q2=v2#FragmentName");

Console.WriteLine($"AbsolutePath: {uri.AbsolutePath}");
Console.WriteLine($"AbsoluteUri: {uri.AbsoluteUri}");
Console.WriteLine($"DnsSafeHost: {uri.DnsSafeHost}");
Console.WriteLine($"Fragment: {uri.Fragment}");
Console.WriteLine($"Host: {uri.Host}");
Console.WriteLine($"HostNameType: {uri.HostNameType}");
Console.WriteLine($"IdnHost: {uri.IdnHost}");
Console.WriteLine($"IsAbsoluteUri: {uri.IsAbsoluteUri}");
Console.WriteLine($"IsDefaultPort: {uri.IsDefaultPort}");
Console.WriteLine($"IsFile: {uri.IsFile}");
Console.WriteLine($"IsLoopback: {uri.IsLoopback}");
Console.WriteLine($"IsUnc: {uri.IsUnc}");
Console.WriteLine($"LocalPath: {uri.LocalPath}");
Console.WriteLine($"OriginalString: {uri.OriginalString}");
Console.WriteLine($"PathAndQuery: {uri.PathAndQuery}");
Console.WriteLine($"Port: {uri.Port}");
Console.WriteLine($"Query: {uri.Query}");
Console.WriteLine($"Scheme: {uri.Scheme}");
Console.WriteLine($"Segments: {string.Join(", ", uri.Segments)}");
Console.WriteLine($"UserEscaped: {uri.UserEscaped}");
Console.WriteLine($"UserInfo: {uri.UserInfo}");

// AbsolutePath: /Home/Index.htm
// AbsoluteUri: https://user:password@www.contoso.com:80/Home/Index.htm?q1=v1&q2=v2#FragmentName
// DnsSafeHost: www.contoso.com
// Fragment: #FragmentName
// Host: www.contoso.com
// HostNameType: Dns
// IdnHost: www.contoso.com
// IsAbsoluteUri: True
// IsDefaultPort: False
// IsFile: False
// IsLoopback: False
// IsUnc: False
// LocalPath: /Home/Index.htm
// OriginalString: https://user:password@www.contoso.com:80/Home/Index.htm?q1=v1&q2=v2#FragmentName
// PathAndQuery: /Home/Index.htm?q1=v1&q2=v2
// Port: 80
// Query: ?q1=v1&q2=v2
// Scheme: https
// Segments: /, Home/, Index.htm
// UserEscaped: False
// UserInfo: user:password
```

- **`System.Uri`**
  https://docs.microsoft.com/en-us/dotnet/api/system.uri?view=net-5.0

WebClient

- Základní třída (fasáda) ulehčující tvorbu kódů, např. možnost přímého přístupu k jiným typům dat, než v podobě streamů
- Jde o zjednodušené využívání tříd `WebRequest` a `WebResponse`
- Některé prvky a funkce nepodporuje (Cookies)
- Umožňuje jak stahování dat (download, GET), tak „zasílání“ dat (upload, POST)
- Podporuje synchronní i asynchronní metody např. `DownloadString` a `DownloadStringAsync/DownloadStringTaskAsync`
- Metody pro download i upload (File, String, Data)

- **`System.Net.WebClient`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.webclient?view=net-5.0

Použití WebClient

```
WebClient wc = new WebClient { Proxy = null };
Uri uri = new Uri("https://www.7timer.info/bin/astro...");

wc.DownloadFileAsync(uri, "weather.txt");
string s = wc.DownloadString(uri);

Console.WriteLine("Number of characters: " + s.Length);
```

```
WebClient wc = new WebClient { Proxy = null };
wc.QueryString.Add("q", "WebClient");      // Search for "WebClient"
wc.QueryString.Add("hl", "en");            // Display page in English
wc.DownloadFile("http://www.google.com/search", "results.html");
```

WebRequest a WebResponse

- Komplexnější a flexibilnější možnosti použití
- Pro přístup k datům se vždy používá stream
- Podle protokolu při vytváření požadavku se využívá konkrétní podtřída `HttpWebRequest`, `FtpWebRequest`, `FileWebRequest` (je možné vytvářet i vlastní)

- **`System.Net.WebRequest`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.webrequest?view=net-5.0
- **`System.Net.WebResponse`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.webresponse?view=net-5.0

Použití WebRequest/WebResponse

```
WebRequest req = WebRequest.Create("http://www...");
req.Proxy = null;

using (WebResponse res = req.GetResponse())
      using (Stream rs = res.GetResponseStream())
            using (FileStream fs = File.Create("code.html"))
                  rs.CopyTo(fs);

WebRequest req = WebRequest.Create("http://www...");
req.Proxy = null;

using (WebResponse res = await req.GetResponseAsync())
      using (Stream rs = res.GetResponseStream())
            using (FileStream fs = File.Create("code.html"))
                  await rs.CopyToAsync(fs);
```

HttpClient

- Nově od .NET 4.5
- Specializuje se na implementaci využívající HTTP protokol s ohledem na přístup k API, REST API, apod.
- Abstrakce nad `HttpWebRequest` a `HttpWebResponse`
- Podporuje
  - více požadavků v rámci jedné instance (na rozdíl od `WebClient`), je tedy žádoucí používat jednu instanci
  - pouze asynchronní metody pro získávání dat
  - možnost tvorby message handlerů pro implemnetaci vlastního zpracování HTTP požadavků/odpovědí. Message handlery je možné řetězit a vytvářet tak specifické procesy zpracování. https://docs.microsoft.com/en-us/aspnet/web-api/overview/advanced/httpclient-message-handlers
  - komplexní systém pro specifikaci hlaviček a obsahu zpráv
  - větší kontrolu nad procesem dotaz/odpověď, např. v případě chyby 404 není vyvolána výjimka, ale o chybu je možné se postarat implementačně

- **`System.Net.Http.HttpClient`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=net-5.0

HttpClient

- `HttpResponseMessage` – třída využívaná jako výsledek požadavku, obsahuje získaných obsah, ale také umožňuje přístup k hlavičkám, kódu odpovědi, atd.
- `GetAsync` – obecná metoda pro získání dat, vracející objekt třídy `HttpResponseMessage`, nevyžaduje předem připravený požadavek (`HttpRequestMessage`)
- `PostAsync`, `PutAsync`, `DeleteAsync` – další obecné metody pro realizaci HTTP příkazů
- `CopyToAsync` – metodu pro propojení streamu s odpovědí a další streamu pro zpracování dat (kopie)

- `HttpRequestMessage` – třída reprezentující požadavek, objekt umožňuje nastavit typ požadavku, pracovat s hlavičkami, autorizaci, atd.
- `SendAsync` – nižší úroveň specifikace požadavku, využívá `HttpRequestMessage`
- `HttpContent` – abstraktní třída umožňující definovat obsah HTTP zpráv, využité např. při uploadu dat v rámci `HttpRequestMessage`

Použití HttpClient

```
string html = await new HttpClient().GetStringAsync("http://www...");

var client = new HttpClient();
var task1 = client.GetStringAsync("http://www...");
var task2 = client.GetStringAsync("http://www...");
Console.WriteLine(await task1);
Console.WriteLine(await task2);

var client = new HttpClient();

HttpResponseMessage response = await client.GetAsync("http://...");
response.EnsureSuccessStatusCode();
string html = await response.Content.ReadAsStringAsync();

var client = new HttpClient();
var request = new HttpRequestMessage(HttpMethod.Get, "http://...");
HttpResponseMessage response = await client.SendAsync(request);
response.EnsureSuccessStatusCode();
```

HttpListener

- Třída umožňující realizaci server-side funkcionality, tj. implementaci HTTP serveru
- Využívá HTTP API hostitelského systému, což umožňuje poslouchat na stejné IP adrese a portu více aplikacím najednou (s ohledem na definovanou adresu a využíván systémového přístupu)
- Pokud je port obsazen aplikací napřímo (pomocí soketů), není možné poslouchat na již takto obsazeném portu
- Pokud je vyžadováno zpracování požadavků souběžně s jinými činnostmi, je nutné realizovat poslouchání v samostatném vlákně (čekání na požadavek od klienta blokuje dané vlákno)

- `HttpListenerContext` – třída reprezentující požadavek ze strany klienta (vlastnost `Request`). Přímo obsahuje také vlastnost `Response` (`HttpListenerResponse`), která slouží ke konstrukci odpovědí na odpovídající konkrétní požadavek.

- **`System.Net.HttpListener`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.httplistener?view=net-5.0

Použití HttpListener

```
HttpListener listener = new HttpListener();
listener.Prefixes.Add("http://localhost:51111/MyApp/");

listener.Start();

// Await a client request:
HttpListenerContext context = await listener.GetContextAsync();

string msg = "You asked for: " + context.Request.RawUrl;
context.Response.ContentLength64 = Encoding.UTF8.GetByteCount(msg);
context.Response.StatusCode = (int)HttpStatusCode.OK;

using (Stream s = context.Response.OutputStream)
      using (StreamWriter writer = new StreamWriter(s))
            await writer.WriteAsync(msg);

listener.Stop();
```

SmtpClient

- Třída umožňující jednoduché zasílání e-mailových zpráv pomocí SMTP protokolu (klientská část)
- Jedná se pouze o zasílání skrz SMTP protokol, protokoly POP3 a IMAP nejsou standardně v .NET zahrnuty a musí se případně implementovat na nižší úrovni (TCP, UDP)
- Je potřeba SMTP server, který přijme požadavek a postará se o další distribuci emailu, viz https://www.youtube.com/watch?v=j7kMZD81hec
- SMTP server obvykle vyžaduje autorizaci pro odesílání e-mailů, zároveň často mívá nastaveno omezení pro přístup (z důvodu zamezení zneužití pro rozesílání SPAMu)
- `MailMessage` – třída umožňující komplexní tvorbu zprávy, včetně příloh, mime-types, atd.

```
SmtpClient client = new SmtpClient();
client.Host = „158.196.100.100";
client.Send("from@adomain.com", "to@adomain.com", "subject", "body");
```

- **`System.Net.Mail.SmtpClient`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.mail.smtpclient?view=net-5.0

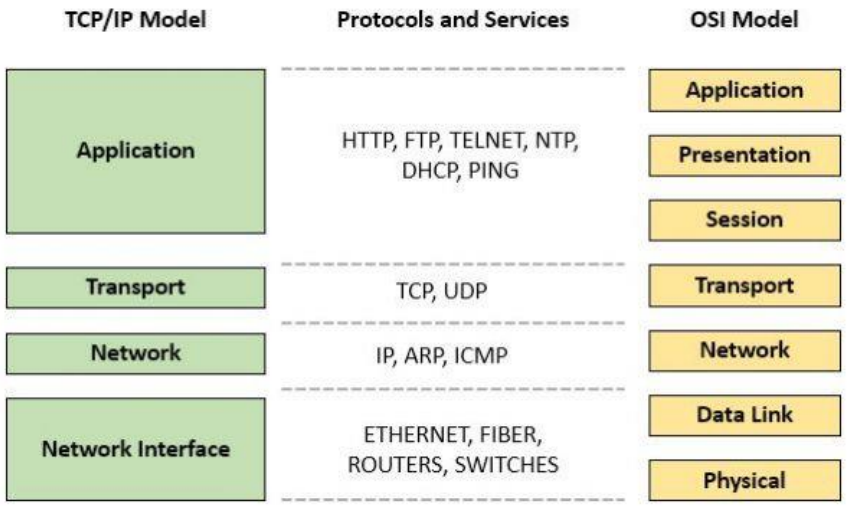
MailKit

- Externí projekt, který poskytuje robustní (dle RFC specifikací) implementaci mailových klientů (POP3, IMAP, SMTP) v rámci .NET
- Umožňuje tedy implementovat nejen alternativní odesílání emailů (SMTP), ale také plnohodnotný přístup k poštovním schránkám (POP3, IMAP)
- Je potřeba využít NuGet balíček **MailKit** https://github.com/jstedfast/MailKit

- https://www.youtube.com/watch?v=bKECC0DHe9Y

```
var message = new MimeMessage();
message.From.Add(new MailboxAddress("Joey Tribbiani", "joey@friends.com"));
message.To.Add(new MailboxAddress("Mrs. Chanandler Bong", "chandler@friends.com"));
message.Subject = "How you doin'?";

message.Body = new TextPart("plain")
    {
        Text = @"Hey Chandler, I just ...?
                 -- Joey"
    };

using (var client = new SmtpClient())
    {
        client.Connect("smtp.friends.com", 587, false);

        // Note: only needed if the SMTP server requires authentication
        client.Authenticate("joey", "password");

        client.Send(message);
        client.Disconnect(true);
    }
```

Protokoly TCP a UDP



- Jedná se o protokoly na transportní vrstvě, které odpovídajícím způsobem řeší samotný přenos dat v síti
- Použití při implementaci vyžaduje nižší míru abstrakce a větší míru technické znalosti.
- Vše, co je realizovatelné třídami `WebClient`, `HttpClient`, `HttpListener`, atd. je realizovatelné na nižší úrovni pomocí tříd `TcpClient`, `TcpListener`, `UdpClient`
- Nicméně se jedná opět o fasádní třídy nad třídou `Socket` – nejnižší úroveň implementace síťové komunikace

- **`System.Net.Sockets`**
  https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets?view=net-5.0

Použití TcpClient a TcpListener

```
using (TcpClient client = new TcpClient())
{
      client.Connect("address", port);
      using (NetworkStream n = client.GetStream())
      {
        // Read and write to the network stream...
      }
}
```

```
TcpListener listener = new TcpListener(< ip address >, port);
listener.Start();

while (keepProcessingRequests)
      using (TcpClient c = listener.AcceptTcpClient())
            using (NetworkStream n = c.GetStream())
                {
                    // Read and write to the network stream...
                }
listener.Stop();
```

Zdroje

- https://docs.microsoft.com/cs-cz/dotnet/
- https://devblogs.microsoft.com/
- https://www.codeguru.com/csharp/

- YAMIKANI FUKIZI, Kenneth, Jason DE OLIVEIRA a Michel BRUCHET. *Learn ASP.NET Core 3: Develop modern web applications*. Second edition. Packt Publishing, 2019. ISBN 978-1789610130.
- ALBAHARI, Joseph. *C# 10.0 in a Nutshell: The Definitive Reference*. O'Reilly Media; 1st edition, 2022. ISBN 978-1098121952.
- ALBAHARI, Joseph. *C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code*. 6th edition. Packt Publishing, 2021. ISBN 978-1801077361.