3 Komunikácia na báze UDP protokolu

UDP (*user datagram protocol*) je jednoduchý protokol bez trvalého pripojenia. Komunikácia je na báze datagramov, ktoré sú vysielané bez spätného overenia, či dorazili do cieľa. Medzi hlavné črty patria:

- **Nespoľahlivosť** - po vyslaní správy nie je možné protokolom overiť, či ju príjemca prijal.
- **Nezoradenosť** - ak sa vyšlú dve správy jednému príjemcovi, nie je možné vopred určiť poradie, v akom dôjdu.
- **Nenáročnosť** - neobsahuje žiadne dodatočné kontrolné správy, sledovanie pripojenia a podobne.
- **Datagramy** - pakety sú vysielané individuálne a kontrolované z hľadiska integrity až po prijatí. Tieto pakety majú definované hranice čo zaručuje, že po prečítaní sa načíta správa vo pôvodnej forme, ako bola vyslaná.
- **Bez kontroly preťaženia** - UDP protokol neobsahuje žiadnu kontrolu preťaženia, tá je možná v aplikačnej úrovni.

3.1 UDP klient - vysielanie

V programovacom jazyku C# sú triedy potrebné pre realizáciu UDP protokolu v namespace System.Net

```
using System.Net;
using System.Net.Sockets;
```

Prvým krokom pred vysielaním je jednorazové zadefinovanie komunikácie na báze UDP protokolu pomocou socketu:

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

Druhým dôležitým krokom je vytvorenie koncového bodu - miesta v sieti, kam bude paket vyslaný. Toto miesto je definované IP adresou a portom:

```
IPAddress IPadresa_ciel = IPAddress.Parse(Console.ReadLine());
int port_ciel = Convert.ToInt32(Console.ReadLine());
IPEndPoint KoncovyBod_ciel = new IPEndPoint(IPadresa_ciel, port_ciel);
```

Podľa typu aplikácie je jednorázovo alebo v slučke načítaná správa, ktorá sa má poslať. Avšak reťazec znakov správy je nutné previesť na pole bajtov, aby ich bolo možné vyslať. Tu treba uvažovať aj kódovanie textu:

```
string sprava = Console.ReadLine();
byte[] buffer = Encoding.UTF8.GetBytes(sprava);
```

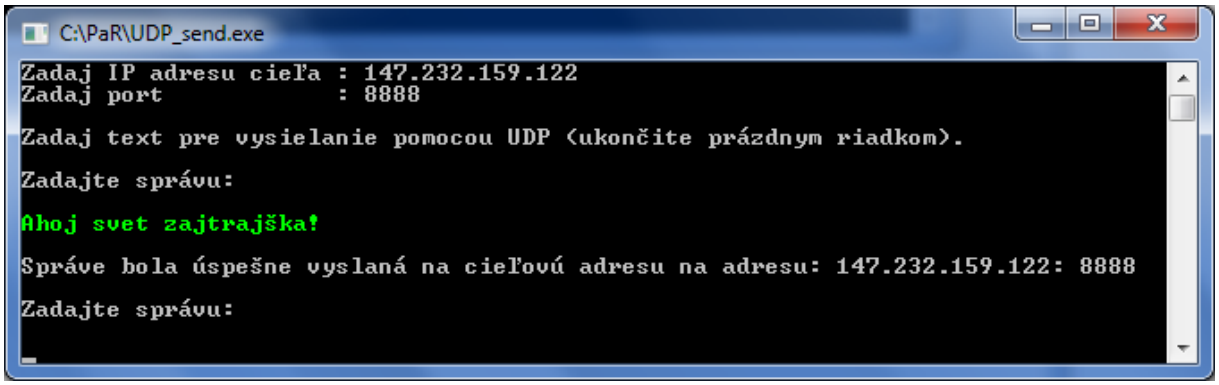
Teraz je možné vyslať datagram na zvolenú adresu. Môže však nastať chyba pre vysielaní na strane klienta, preto je vhodne ošetrenie pre výnimky:

```
try
{
  socket.SendTo(buffer, KoncovyBod_ciel);
}
catch (Exception ex)
{
  chyba = true;
  Console.WriteLine("Doslo k chybe {0}", ex.Message);
}
```

Ak nedošlo k chybe, správa bola úspešne vyslaná na požadovanú adresu a port. Ďalej je uvedený výstup aplikácie, ktorá realizuje UDP klient vo forme konzolovej aplikácie:

3.2 UDP server - prijímanie

Prijímač UDP vysielania by mal bežať nezávisle od vysielača - buď ako samostatná aplikácia alebo vlákno. Aby bolo možné vypísať IP adresu prijímača, je vhodné napísať metódu:



Obr. 1: Príklad konzolového klienta pre UDP - vysielanie

```
    /// <summary>
    /// Zisti moju IP adresu v sieti
    /// </summary>
    /// <returns>IP adresa</returns>
    public static string ZistiMojuIPadresu()
    {
        IPHostEntry host;
        string localIP = "?";
        host = Dns.GetHostEntry(Dns.GetHostName());
        foreach (IPAddress ip in host.AddressList)
        {
            if (ip.AddressFamily.ToString() == "InterNetwork")
            {
                localIP = ip.ToString();
            }
        }
        return localIP;
    }
```

Znalosť vlastnej IP adresy nie je pre príjem správy kľúčová, nakoľko sleduje sa hlavne port s konkrétnym číslom, na ktorý môže byť vyslaná správa z iného zdroja:

```
Console.Write("Zadaj port: ");
int port_prijem = Convert.ToInt32(Console.ReadLine());
```

Na základe tohto portu je možné definovať vlastný koncový bod (skupinu koncových bodov) a to ako:

```
UdpClient prijimac = new UdpClient(port_prijem);
IPEndPoint skupinaKB = new IPEndPoint(IPAddress.Any, port_prijem);
```

Teraz nasleduje jednorázové alebo opakované čakanie na príjem správy. Správa bude prijatá vo forme bytov a aby ju bolo možné vypísať ako reťazec, je nutné vykonať prevod s uvažovaním správneho kódovania. Argumentom metódy *Recieve* je referencia na skupinu koncových bodov:

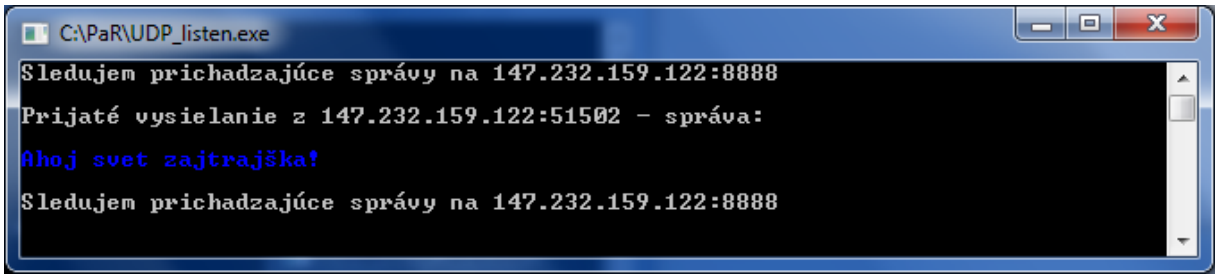
```
byte[] buffer_prijaty = prijimac.Receive(ref skupinaKB);
string sprava_prijata = Encoding.UTF8.GetString(buffer_prijaty, 0, buffer_prijaty.Length);
Console.ForegroundColor = ConsoleColor.Blue;
Console.WriteLine(sprava_prijata);
```

Vykonávanie programu a metódy však stojí na príjme prvej správy a preto je problematické prepojiť UDP klient a UDP vysielač - vyžaduje to nezávislé aplikácie alebo samostatné vlákna. Ďalej je uvedený výstup aplikácie, ktorá realizuje UDP server vo forme konzolovej aplikácie:

3.3 Úloha pre riešenie na cvičení

Naprogramujte Windows Forms aplikáciu, ktorá umožňuje:

- Vie vyslať správu na používateľom zvolenú IPv4 adresu a port (hlavná úloha)
- Obsahuje ošetrenie vstupu IP adresy, portu a správy



Obr. 2: Príklad konzolového klienta pre UDP - prijímanie správ

- Pridáva pred správu prefix vo forme času, používateľa a IP adresy.

```
string prefix = "[10:12:33] - Pouzivatel - 147.232.159.101 - ";
```

- Vie prijať správu na používateľom zvolenom porte (pomocou vlákna)
- Umožňuje zapínať/vypínať príjem správ