Standardy programowania protokołów komunikacyjnych

Laboratorium nr 5 – komunikacja multicastowa IPv6

Celem ćwiczenia jest zdobycie umiejętności programowania komunikacji multicastowej za pomocą protokołu IPv6 w języku C# na poziomie interfejsu gniazd. Materiały dotyczące poszczególnych funkcji były podane na wykładzie.

1. Narzędzia i konfiguracja

W systemach wbudowanych użyte zostanie środowisko .Net Core i język programowania C# https://dotnet.microsoft.com/download.

Zalecanym środowiskiem deweloperskim jest Visual Studio Code https://code.visualstudio.com/download.

```
dotnet new console -o nazwa
cd nazwa
#edycja Program.cs
code .
#uruchomienie w srodowisku developerskim
dotnet run
#publikacja gotowej aplikacji
dotnet publish -r linux-arm
#skopiować całą zawartość folderu publish do Pi i nadać atrybut x aplikacji
```

W celu uruchomienia aplikacji w systemie Pi należy skopiować zawartość folderu publish do Raspberry Pi. Alternatywnie można pliki źródłowe umieścić w systemie Raspberry Pi i w tym systemie wykonywać kompilację i uruchomienie programów.

2. Funkcje niezbędne do obsługi ruchu multicastowego

Aby możliwe było odbieranie datagramów adresowanych do grup multicastowcy należy dołączyć do odpowiedniej grupy multicastowej. Realizowane jest to za pomocą funkcji **`JoinMulticastGroup`**`()` i **`DropMulticastGroup`**().

3. Multicastowy nadajnik datagramów UDP

Program wysyła na adres multicastowy datagram UDP z komunikatem zawierającym tekstową wersję lokalnego czasu. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametry, docelowy adres multicastowy i numer portu. Należy pamiętać, że adresy multicastowe IPv6 powinny zawierać numer interfejsu (ang. *ScopeID*) przez, który mają być wysyłane.

mcastcli.cs

```
1  using System;
2  using System.Net;
3  using System.Net.Sockets;
4  using System.Text;
5
6  namespace mcastcli
7  {
       0 references
8      class MainClass
9      {
           0 references
10         public static void Main(string[] args)
11         {
12             string group = "ff12::1%1";
13             int port = 8888;
14             string data = "Test data. " + DateTime.Now.ToString();
15
16             if (args.Length > 0) group = args[0];
17             if (args.Length > 1) port = int.Parse(args[1]);
18             if (args.Length > 2) data = args[2];
19
20             IPAddress[] addr = Dns.GetHostAddresses(group);
21             UdpClient udpClient = new UdpClient(addr[0].AddressFamily);
22             udpClient.Client.ReceiveTimeout = 3000;
23             if (addr[0].IsIPv6Multicast)
24                 udpClient.JoinMulticastGroup((int)addr[0].ScopeId, addr[0]);
25             else if (addr[0].AddressFamily == AddressFamily.InterNetwork)
26                 udpClient.JoinMulticastGroup(addr[0]);
27             Console.WriteLine($"Joined multicast group {addr[0]}");
28             try
29             {
30                 byte[] bufout = Encoding.UTF8.GetBytes(data);
31                 udpClient.Send(bufout, bufout.Length, new IPEndPoint(addr[0], port));
32                 Console.WriteLine($"Sent to: {addr[0]} data: {data}");
33                 IPEndPoint remoteEP = new IPEndPoint(IPAddress.IPv6Any, 0);
34                 byte[] bufin = udpClient.Receive(ref remoteEP);
35                 data = Encoding.UTF8.GetString(bufin);
36                 Console.WriteLine($"Received response from: {remoteEP.Address} data: {data}");
37             }
38             catch
39             {
40                 Console.WriteLine("No response received");
41             }
42             if (addr[0].IsIPv6Multicast)
43                 udpClient.DropMulticastGroup(addr[0], (int)addr[0].ScopeId);
44             else if (addr[0].AddressFamily == AddressFamily.InterNetwork)
45                 udpClient.DropMulticastGroup(addr[0]);
46             Console.WriteLine($"Disconnected from multicast group {addr[0]}");
47             udpClient.Close();
48             Console.WriteLine("Closed UDP socket");
49         }
50     }
51 }
52
```

4. Multicastowy odbiornik datagramów UDP

Program nasłuchuje na adresie multicastowym i wypisuje treści komunikatów przesłanych datagramami UDP. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametry, źródłowy adres multicastowy i numer portu. Należy pamiętać, że adresy multicastowe IPv6 powinny zawierać numer interfejsu (ang. *ScopeID*) przez, który mają być wysyłane.

mcastsrv.cs > ...

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

namespace mcastsrv
{
    0 references
    class MainClass
    {
        0 references
        public static void Main(string[] args)
        {
            string group = "ff12::1%1";
            int port = 8888;

            if (args.Length > 0) group = args[0];
            if (args.Length > 1) port = int.Parse(args[1]);

            IPAddress[] addr = Dns.GetHostAddresses(group);
            UdpClient udpServer = new UdpClient(port, addr[0].AddressFamily);
            Console.WriteLine($"UDP multicast server start at address {addr[0]} port {port}");
            try
            {
                if (addr[0].IsIPv6Multicast)
                    udpServer.JoinMulticastGroup((int)addr[0].ScopeId, addr[0]);
                else if (addr[0].AddressFamily == AddressFamily.InterNetwork)
                    udpServer.JoinMulticastGroup(addr[0]);
                Console.WriteLine($"Joined multicast group {addr[0]}");
                while (true)
                {
                    IPEndPoint remoteEP = new IPEndPoint(IPAddress.IPv6Any, 0);
                    byte[] bufin = udpServer.Receive(ref remoteEP);
                    string data = Encoding.UTF8.GetString(bufin);
                    Console.WriteLine($"Received from: {remoteEP.Address} data: {data}");
                    byte[] bufout = Encoding.UTF8.GetBytes("Echo: " + data);
                    udpServer.SendAsync(bufout, bufout.Length, remoteEP);
                }
            }
            finally
            {
                if (addr[0].IsIPv6Multicast)
                    udpServer.DropMulticastGroup(addr[0], (int)addr[0].ScopeId);
                else if (addr[0].AddressFamily == AddressFamily.InterNetwork)
                    udpServer.DropMulticastGroup(addr[0]);
                Console.WriteLine("Droped multicast group {addr[0]}");
                udpServer.Close();
                Console.WriteLine("Closed socket");
            }
        }
    }
}
```