Oprogramowanie komunikacyjne dla Internetu rzeczy

Laboratorium nr 4 – komunikacja unicastowa IPv6

Celem ćwiczenia jest zdobycie umiejętności programowania komunikacji unicastowej 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. Najważniejsze struktury danych i najważniejsze funkcje

2.1.Struktury IPAddress i IPEndPoint

Reprezentacją adresów IP w środowisku .Net są klasy `System.Net.IPAddress` i `System.Net.EndPoint`. Klasa IPEndPoint zawiera informacje zarówno o adresie jak i porcie połączenia. Obie klasy są wspólne dla rodzin protokołów IP i IPv6.

2.4 Obsługa IPv6 w języku C#

Poprawną konwersję zapisów tekstowych adresów domenowych albo adresów IP na klasę IPAddress zapewnia metoda **`IPAddress[]`** `System.Net.`**`Dns.GetHostAddresses(`**`string host`**`)`. Zwraca ona tablicę wszystkich adresów przypisanych do danej nazwy. Kolejność adresów wynika z preferencji systemu (obecnie zawsze najpierw IPv6). Jeśli w systemie nie ma obsługi IPv6 albo połączenie jest nieaktywne zwracana lista nie zawiera adresów IPv6.

Utworzenie gniazda dla IPv6 wymaga podania jako parametru `System.Net.Sockets.`**`AddressFamily.InterNetworkV6`**.

Przykłady:

```
var udpClient = new UdpClient(AddressFamily.InterNetworkV6);
var udpServer = new UdpClient(AddressFamily.InterNetworkV6);

var tcpClient = new TcpClient(AddressFamily.InterNetworkV6);
var tcpServer = new TcpListener(IPAddress.IPv6Any, port);
```

W SO Windows NT6 wprowadzono funkcjonującą wcześniej w SO Linux funkcję umożliwiającą obsługę przychodzących połączeń zarówno IPv4 jak i IPv6 za pomocą pojedynczego gniazda IPv6 z wyzerowaną opcją IPV6ONLY. Dla środowiska .Net można taką obsługę włączyć ustawiając właściwość DualMode na wartość true. Przychodzące połączenia IPv4 mają w tym rozwiązaniu adresy mapowane do IPv6 w postaci **`::ffff:1.2.3.4`**.

Przykład inicjacji serwera TCP do równoczesnej obsługi połączeń IPv4 i IPv6:

```
Var tcpsrv = new TcpListener(IPAddress.IPv6Any, port);
tcpsrv.Server.DualMode = true;
tcpsrv.Start();
```

Przykład inicjacji serwera UDP do równoczesnej obsługi połączeń IPv4 i IPv6:

```
var udpsrv = new UdpClient(AddressFamily.InterNetworkV6);
udpsrv.Client.DualMode = true;
udpsrv.Client.Bind(new IPEndPoint(IPAddress.IPv6Any, port));
```

3. Odczytanie wszystkich adresów lokalnych interfejsów

Program umożliwia wypisanie wszystkich adresów lokalnych interfejsów. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametr, adres domenowy, dla którego zostaną zwrócone wszystkie adresy IP.

Uwaga! Program nie zawiera obsługi błędów.

addrs.cs > ...

```
1  using System;
2  using System.Net;
3  using System.Net.NetworkInformation;
4  using System.Net.Sockets;
5
6  namespace addrs
7  {
       0 references
8      class Program
9      {
           0 references
10         static void Main(string[] args)
11         {
12             string host = Dns.GetHostName();
13             if (args.Length > 0) host = args[0];
14
15             Console.WriteLine("Local interfaces:");
16             foreach (var iface in NetworkInterface.GetAllNetworkInterfaces())
17             {
18                 Console.WriteLine($"Interface: {iface.Name}");
19                 foreach (var ip in iface.GetIPProperties().UnicastAddresses)
20                 {
21                     if (ip.Address.AddressFamily == AddressFamily.InterNetworkV6)
22                         Console.Write(" Address IPv6 ");
23                     else
24                         Console.Write(" Address IPv4 ");
25                     Console.WriteLine(ip.Address);
26                 }
27             }
28             Console.WriteLine();
29
30             Console.WriteLine($"Hostname: {host}");
31             foreach (var addr in Dns.GetHostAddresses(host))
32             {
33                 if (addr.AddressFamily == AddressFamily.InterNetworkV6)
34                     Console.Write(" Address IPv6: ");
35                 else
36                     Console.Write(" Address IPv4: ");
37                 Console.WriteLine(addr);
38             }
39         }
40     }
41 }
42
```

4. Klient połączenia TCP

Program umożliwia wysłanie tekstu protokołem TCP i odebrania odpowiedzi w ramach tego samego połączenia. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametry, adres, numer portu i treść wiadomości.

Uwaga! Program nie zawiera obsługi błędów.

tcpcli.cs > ...

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

namespace tcpcli
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string host = "ipv6.google.com";
            int port = 80;
            string mesg = "GET / HTTP/1.1\r\n\r\n";
            int BUFSIZE = 1500;
            char[] buf = new char[BUFSIZE];

            if (args.Length > 0) host = args[0];
            if (args.Length > 1) port = int.Parse(args[1]);
            if (args.Length > 2) mesg = args[2];

            IPAddress[] addr = Dns.GetHostAddresses(host);
            TcpClient tcp = new TcpClient(addr[0].AddressFamily);
            tcp.Client.ReceiveTimeout = 3000;
            tcp.Connect(addr[0], port);
            Console.WriteLine($"Connected to: {addr[0]}");
            using (var streamWriter = new StreamWriter(tcp.GetStream(), Encoding.UTF8))
            using (var streamReader = new StreamReader(tcp.GetStream(), Encoding.UTF8))
            {
                streamWriter.Write(mesg);
                streamWriter.Flush();
                Console.WriteLine($"Sent to: {addr[0]} data: {mesg}");
                System.Threading.Thread.Sleep(100);
                int len = streamReader.Read(buf, 0, BUFSIZE);
                if (len > 0)
                {
                    Console.WriteLine($"Received response length: {len}");
                    string sin = new string(buf, 0, len);
                    Console.WriteLine(sin);
                }
            }
            tcp.Close();
            Console.WriteLine("Closed connection");
        }
    }
}

```

5. Echo serwer dla połączeń TCP

Program przyjmuje połączenia i odsyła z powrotem dane przesłane protokołem TCP w ramach tego samego połączenia. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametr, numer portu.

Uwaga! Program nie zawiera obsługi błędów.

tcpsrv.cs > ...

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

namespace tcpsrv
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int port = 8000;
            if (args.Length > 0) port = int.Parse(args[0]);

            int BUFSIZE = 1500;
            char[] buf = new char[BUFSIZE];
            TcpListener tcp = new TcpListener(IPAddress.IPv6Any, port);
            tcp.Server.DualMode = true;
            tcp.Start();
            Console.WriteLine($"TCP Server started at port {port}");
            while (true)
            {
                TcpClient client = tcp.AcceptTcpClient();
                IPEndPoint remoteIPEndPoint = (IPEndPoint)client.Client.RemoteEndPoint;
                string remoteIP;
                if (remoteIPEndPoint.Address.IsIPv4MappedToIPv6)                //::ffff:1.2.3.4
                    remoteIP = remoteIPEndPoint.Address.MapToIPv4().ToString();
                else
                    remoteIP = remoteIPEndPoint.Address.ToString();
                using (var streamReader = new StreamReader(client.GetStream(), Encoding.UTF8))
                using (var streamWriter = new StreamWriter(client.GetStream(), Encoding.UTF8))
                {
                    int size = streamReader.Read(buf, 0, BUFSIZE);
                    if (size > 0)
                    {
                        string data = new String(buf, 0, size);
                        Console.WriteLine($" Received from: {remoteIP} data: {data}");
                        streamWriter.Write(data);
                        streamWriter.Flush();
                        Console.WriteLine(" Sent echo response");
                    }
                }
                client.Close();
                Console.WriteLine("Closed client connection");
            }
            tcp.Stop();
            Console.WriteLine("Closed TCP server");
        }
    }
}
```

6. Klient komunikacji UDP

Program umożliwia wysłanie tekstu protokołem UDP i odebrania odpowiedzi w ramach tego samego „połączenia". Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając z wiersza poleceń można podać, jako parametry, adres, numer portu i treść wiadomości.

Uwaga! Program nie zawiera obsługi błędów.

udpcli.cs > ...

```
1  using System;
2  using System.Net;
3  using System.Net.Sockets;
4  using System.Text;
5
6  namespace udpcli
7  {
       0 references
8      class Program
9      {
           0 references
10         static void Main(string[] args)
11         {
12             string host = "localhost";
13             int port = 8000;
14             string mesg = "Test message " + DateTime.Now.ToString();
15
16             if (args.Length > 0) host = args[0];
17             if (args.Length > 1) port = int.Parse(args[1]);
18             if (args.Length > 2) mesg = args[2];
19
20             IPAddress[] addr = Dns.GetHostAddresses(host);
21             IPEndPoint remoteEP = new IPEndPoint(IPAddress.IPv6Any, 0);
22             UdpClient udpClient = new UdpClient(addr[0].AddressFamily);
23             udpClient.Client.ReceiveTimeout = 3000;
24             try
25             {
26                 udpClient.Connect(addr[0], port);
27                 byte[] buf = Encoding.UTF8.GetBytes(mesg);
28                 udpClient.Send(buf, buf.Length);
29                 Console.WriteLine($"Sent to: {addr[0]} data: {mesg}");
30                 byte[] bufin = udpClient.Receive(ref remoteEP);
31                 if (bufin.Length > 0)
32                 {
33                     string resp = Encoding.UTF8.GetString(bufin);
34                     Console.WriteLine($"Received from: {remoteEP.Address} data: {resp}");
35                 }
36             }
37             catch
38             {
39                 Console.WriteLine("No response received");
40             }
41             udpClient.Close();
42         }
43     }
44 }
45
```

7. Echo serwer dla komunikacji UDP

Program przyjmuje przychodzące datagramy UDP i odsyła z powrotem dane przesłane protokołem UDP w ramach tego samego „połączenia”. Program obsługuje zarówno protokół IPv6 jak i IPv4. Uruchamiając w wierszu poleceń można podać, jako parametr, numer portu.

Uwaga! Program nie zawiera obsługi błędów.

udpsrv.cs > ...

```
1  using System;
2  using System.Net;
3  using System.Net.Sockets;
4  using System.Text;
5
6  namespace udpsrv
7  {
       0 references
8      class Program
9      {
           0 references
10         static void Main(string[] args)
11         {
12             int port = 8000;
13             if (args.Length > 0) port = int.Parse(args[0]);
14
15             UdpClient udpServer = new UdpClient(AddressFamily.InterNetworkV6);
16             udpServer.Client.DualMode = true;
17             try
18             {
19                 udpServer.Client.Bind(new IPEndPoint(IPAddress.IPv6Any, port));
20                 Console.WriteLine($"UDP server started at port {port}");
21                 while (true)
22                 {
23                     IPEndPoint remoteEP = new IPEndPoint(IPAddress.IPv6Any, 0);
24                     byte[] buffer = udpServer.Receive(ref remoteEP);
25                     string data = Encoding.UTF8.GetString(buffer);
26                     Console.WriteLine($"Received from: {remoteEP} Data: {data}");
27                     udpServer.Send(buffer, buffer.Length, remoteEP);
28                     Console.WriteLine($"Sent echo response to: {remoteEP}");
29                 }
30             }
31             finally
32             {
33                 udpServer.Close();
34             }
35         }
36     }
37 }
38
```