A Mutable Log

A blog by Devendra Tewari

Project maintained by tewarid

Hosted on GitHub Pages — Theme by mattgraham

Read ICMP packets in C# using raw sockets

Raw sockets are useful when you want to read all details of a network packet such as the IP header, protocol header, and payload. They can also be used to communicate using protocols that don't have a higher level API, which means most protocols that are not UDP or TCP based. Reading ICMP protocol packets is one useful example.

The following source code is a console application. It receives any kind of IP version 4 ICMP protocol packet, from any host, and prints the entire packet to console.

```
static class Program
{
    private static Socket icmpSocket;
    private static byte[] receiveBuffer = new byte[256];
    private static EndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, 0);

    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    static void Main()
    {
        CreateIcmpSocket();
        while (true) { Thread.Sleep(10); }
    }

    private static void CreateIcmpSocket()
    {
        icmpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Raw, Proto
        icmpSocket.Bind(new IPEndPoint(IPAddress.Any, 0));
        // Uncomment to receive all ICMP message (including destination unreachab
        // Requires that the socket is bound to a particular interface. With mono
        // fails on any OS but Windows.
        //if (Environment.OSVersion.Platform == PlatformID.Win32NT)
        //{
```

```
            BeginReceiveFrom();
        }

        private static void BeginReceiveFrom()
        {
            icmpSocket.BeginReceiveFrom(receiveBuffer, 0, receiveBuffer.Length, Socke
                ref remoteEndPoint, ReceiveCallback, null);
        }

        private static void ReceiveCallback(IAsyncResult ar)
        {
            int len = icmpSocket.EndReceiveFrom(ar, ref remoteEndPoint);
            Console.WriteLine(string.Format("{0} Received {1} bytes from {2}",
                DateTime.Now, len, remoteEndPoint));
            LogIcmp(receiveBuffer, len);
            BeginReceiveFrom();
        }

        private static void LogIcmp(byte[] buffer, int length)
        {
            for (int i = 0; i < length; i++)
            {
                Console.Write(String.Format("{0:X2} ", buffer[i]));
            }
            Console.WriteLine("");
        }
    }
```

Here's an example of the kind of output the program spits

```
26/11/2013 16:42:40 Received 56 bytes from 192.168.61.61:0
45 00 00 38 01 36 00 00 40 01 7D C3 C0 A8 3D 3D C0 A8 3D 3E 04 00 A7 61 00 00 00
 00 45 00 00 27 31 4F 00 00 80 11 FF 22 C0 A8 3D 3E 0C 00 00 6E E8 9E 0F E5 00 1
3 5C 07
26/11/2013 16:58:02 Received 60 bytes from 192.168.61.61:0
45 00 00 3C 01 A6 00 00 40 01 7D 4F C0 A8 3D 3D C0 A8 3D 3E 03 06 9B CF 00 00 00
 00 46 00 00 28 68 6F 00 00 01 02 DE 63 C0 A8 3D 3E E0 00 00 16 94 04 00 00 22 0
0 F9 01 46 00 00 28
```

The first 20 bytes are the IP v4 header, followed by the ICMP payload. The first byte of ICMP payload identifies the type of the ICMP packet. For instance, a value of 3 indicates that the host is

0 Comments - *powered by utteranc.es*

Write Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub