Lab 2: Packet Capture (Filtering)

Details

Aim: To provide an understanding of events in reading data packets

Activities

Using the previous solution from Lab 1, update with the following code [1]. In this case the 2nd connection is used (getNetConnections[1]) in a promiscuous mode (change, as required, depending on your network connection). USE THE CONNECTION WHICH IS THE ETHERNET CONNECTION.

↳http://www.dcs.napier.ac.uk/~bill/WinPCap2.zip

```
using System;
using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace NapierCapture
{
   public class CapturePackets
   {
      public static void Main(string[] args)
      {
         PcapDeviceList getNetConnections = SharpPcap.GetAllDevices();

         // network connection 1 (change as required)
         NetworkDevice netConn = (NetworkDevice)getNetConnections[1];
         PcapDevice device = netConn;

         // Define packet handler
         device.PcapOnPacketArrival +=
            new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);

         //Open the device for capturing
         //true -- means promiscuous mode
         //1000 -- means a read wait of 1000ms
         device.PcapOpen(true, 1000);

         Console.WriteLine("Network connection: {0}",device.PcapDescription);

         //Start the capturing process
         device.PcapStartCapture();

         Console.Write("Press any <RETURN> to exit");
         Console.Read();

         device.PcapStopCapture();
         device.PcapClose();
      }
      private static void device_PcapOnPacketArrival(object sender, Packet packet)
      {
         DateTime time = packet.PcapHeader.Date;
         int len = packet.PcapHeader.PacketLength;
         Console.WriteLine("{0}:{1}:{2},{3} Len={4}",time.Hour, time.Minute,
                     time.Second, time.Millisecond, len);
      }
   }
}
```

Run the program, and produce some network traffic and versify that it is capturing packets, such as:

```
13:17:56,990 Len=695
13:17:57,66 Len=288
13:17:57,68 Len=694
13:18:4,363 Len=319
13:18:4,364 Len=373
13:18:4,364 Len=371
13:18:4,365 Len=375
13:18:4,366 Len=367
```

Did it capture packets? **Yes/No**

Update the code with a filter. In the following case an **IP** and **TCP** filter is used [1]:

```
device.PcapOpen(true, 1000);

Console.WriteLine("Network connection: {0}",device.PcapDescription);

string filter = "ip and tcp";

//Associate the filter with this capture
device.PcapSetFilter( filter );

//Start the capturing process
device.PcapStartCapture();
```

Generate some data traffic, such as loading a Web page, and show that the program is capturing the data packets.

Did it capture packets? **Yes/No**

Next update the filter so that it only captures **ICMP** packets, such as:

```
string filter = "icmp";
```

Generate some data traffic, and prove that it does not capture the packets. Now ping a node on your network, such as:

```
Ping 192.168.1.102
```

And prove that it captures the data packets, such as:

```
13:40:47,761 Len=74
13:40:48,756 Len=74
13:40:48,759 Len=74
13:40:49,757 Len=74
13:40:49,760 Len=74
13:40:50,757 Len=74
```

Did it capture ICMP packets? **Yes/No**

[1] This code is based on the code wrapper for WinPCap developed by T.Gal [http://www.thecodeproject.com/csharp/sharppcap.asp].