コンピュータのネットワークカードの名前の一覧を列挙

まぁ、2.0 ならこれで一覧がとれるかな？と。

```
using System.Net.NetworkInformation;


NetworkInterface[] nic = NetworkInterface.GetAllNetworkInterfaces();
foreach (NetworkInterface var in nic)
{
    Console.WriteLine(var.Description);
}
```

「MS TCP Loopback interface」 ってやつは、リストアップする際に除いたほうがイイ。
調べたら出てくると思うけど、localhost 接続用の仮想 NIC だったような気がする。たしか・・・。

WMI を使って、NIC 一覧を取得するなら、こんな感じ。

```
using System.Management; // 参照設定に System.Management を追加


ManagementClass mc = new ManagementClass("Win32_PerfRawData_Tcpip_NetworkInterface");
ManagementObjectCollection moc = mc.GetInstances();

foreach (ManagementObject mo in moc)
{
    // 情報を表示
    Console.WriteLine("名前     = {0}", mo["Name"]);
    Console.WriteLine("接続速度 = {0} Mbps", Convert.ToInt32(mo["CurrentBandwidth"]) / 1000 / 1000);


    Console.WriteLine("------");
}
```

無効になってる NIC は拾えないので（今は未検証だけど、たしかそうだった)、ご注意。

マジックパケットの送信（Wake On Lan、WOL）

遠隔からコンピュータを起動する機能です。
詳しくは下記 URL を参照。
http://www.atmarkit.co.jp/fwin2k/win2ktips/715wol/wol.html

ソースコード

オーバーロードしてるから、長く見えるけど、要は一番下のメソッドが重要。

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


/// <summary>
/// マジックパケットを送信します。
/// </summary>
/// <param name="address">IP アドレス </param>
/// <param name="subnetmask"> サブネットマスク </param>
/// <param name="physicalAddress"> 起動するマシンの MAC アドレス </param>
private void SendMagicPacket(string address, string subnetmask, byte[] physicalAddress)
{
    SendMagicPacket(IPAddress.Parse(address), IPAddress.Parse(subnetmask), physicalAddress);
```

```
}

/// <summary>
/// マジックパケットを送信します。
/// </summary>
/// <param name="address">IP アドレス </param>
/// <param name="subnetmask"> サブネットマスク </param>
/// <param name="physicalAddress"> 起動するマシンの MAC アドレス </param>
private void SendMagicPacket(IPAddress address, IPAddress subnetmask, byte[] physicalAddress)
{
    uint uip = BitConverter.ToUInt32(address.GetAddressBytes(), 0);
    uint usub = BitConverter.ToUInt32(subnetmask.GetAddressBytes(), 0);

    uint result = uip | (usub ^ 0xFFFFFFFF);

    SendMagicPacket(new IPAddress(result), physicalAddress);
}

/// <summary>
/// ブロードキャストアドレス（255.255.255.255）に対してマジックパケットを送信します。
/// </summary>
/// <param name="physicalAddress"> 起動するマシンの MAC アドレス </param>
private void SendMagicPacket(byte[] physicalAddress)
{
    SendMagicPacket(IPAddress.Broadcast, physicalAddress);
}

/// <summary>
/// 指定されたアドレスに対してマジックパケットを送信します。
/// 送信先のアドレスはブロードキャストアドレスである必要があります。
/// </summary>
/// <param name="broad"> ブロードキャストアドレス </param>
/// <param name="physicalAddress"> 起動するマシンの MAC アドレス </param>
private void SendMagicPacket(string broad, byte[] physicalAddress)
{
    SendMagicPacket(IPAddress.Parse(broad), physicalAddress);
}

/// <summary>
/// 指定されたアドレスに対してマジックパケットを送信します。
/// 送信先のアドレスはブロードキャストアドレスである必要があります。
/// </summary>
/// <param name="broad"> ブロードキャストアドレス </param>
/// <param name="physicalAddress"> 起動するマシンの MAC アドレス </param>
private void SendMagicPacket(IPAddress broad, byte[] physicalAddress)
{
    MemoryStream stream = new MemoryStream();
    BinaryWriter writer = new BinaryWriter(stream);
    for (int i = 0; i < 6; i++)
    {
        writer.Write((byte)0xff);
    }
    for (int i = 0; i < 16; i++)
    {
        writer.Write(physicalAddress);
    }

    UdpClient client = new UdpClient();
    client.EnableBroadcast = true;
    client.Send(stream.ToArray(), (int)stream.Position, new IPEndPoint(broad, 0));
}
```

UDP 通信を Socket で行う。TTL の指定 & ブロードキャスト。

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

普通

```
// 送信先
IPEndPoint remoteIP = new IPEndPoint(IPAddress.Parse("192.168.11.2"), 80);

// 送信データ
byte[] data = new byte[16];

// UDP ソケットの作成
```

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

// TTL を設定
// TTL とは…→  http://e-words.jp/w/TTL-2.html
s.SetSocketOption(SocketOptionLevel.IP, SocketOptionName.IpTimeToLive, 255);

// データを送信
s.SendTo(data, 0, data.Length, SocketFlags.None, remoteIP);
```

ブロードキャスト

```
IPEndPoint remoteIP = new IPEndPoint(IPAddress.Broadcast, 10002);

byte[] data = new byte[16];
Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
s.SetSocketOption(SocketOptionLevel.IP, SocketOptionName.IpTimeToLive, 16);
// ブロードキャストはこれで許可を入れないといけないっぽい
s.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

s.SendTo(data, data.Length, SocketFlags.None, remoteIP);
```

TCP 接続 を Socket クラスで行う（同期通信）

```
using System.Net;
using System.Net.Socket;


// 接続先の IP アドレス
IPAddress addr = IPAddress.Parse("192.168.10.227");
// 接続先ポート
int port = 10001;
// 接続先情報を作る
IPEndPoint end = new IPEndPoint(addr, port);

// TCP ソケットを作る
Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
// タイムアウトを設定
socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.SendTimeout, 1000);
socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, 1000);

// 接続
socket.Connect(end);

// ソケットからストリームを作る
NetworkStream stream = new NetworkStream(socket);
```

後は、このストリームを読み書きすればなんとかなる。

つか、ソケット通信周りは、MSDN でも見てください。

http://msdn2.microsoft.com/ja-jp/library/w89fhyex.aspx

MAC アドレスの取得

MAC アドレスは全部で６バイトある。
最初の３バイトが各メーカーに割り当ててあり、あとの３バイトはメーカーが独自に割り当てる。らしい。
http://www.coffer.com/mac_find/　ここに、MAC アドレスから、そのアドレスを使ってるメーカーを検索して表示するサイトがある。

SendARP を使ったサンプル

このサンプルは ARP とかいうプロトコルを使って、対象の MAC アドレスを問い合わせる。
なので、このコードの場合、ネットワークと通信出来ないと MAC アドレスを取得できない。

```
using System;
using System.Net;
using System.Runtime.InteropServices;


[DllImport("iphlpapi.dll", ExactSpelling=true)]
private static extern int SendARP( int DestIP, int SrcIP, byte[] pMacAddr, ref int PhyAddrLen );

private byte[] getMacAddress(string val){
    return getMacAddress(IPAddress.Parse(val));
}

private byte[] getMacAddress(IPAddress addr)
{
    byte[] mac = new byte[6];
    int len = mac.Length;
    int r = SendARP( BitConverter.ToInt32(addr.GetAddressBytes(), 0), 0, mac, ref len );

    return mac;
}
```

WMI を使って取得。

こっち のソースコード見ちゃってください。

NetworkInformation を使って取得（2.0 用）

以下のようなコードで、コンソールに MAC アドレスの一覧が出力できる。

```
using System.Net.NetworkInformation;

NetworkInterface[] nics = NetworkInterface.GetAllNetworkInterfaces();
foreach (NetworkInterface var in nics)
{
    PhysicalAddress phy = var.GetPhysicalAddress();
    Console.WriteLine(phy);
}
```

接続してきた相手の IP アドレスを取得。

```
// TcpListener で接続をうけ、Socket として取る。
Socket client = tcplistener.AcceptSocket();

// エンドポイントとかいうのをもらう
IPEndPoint endpoint = (IPEndPoint)client.RemoteEndPoint;

// そこから接続している相手の IPAddress が取れる。
IPAddress address = endpoint.Address;

// NetworkStream を作成。
NetworkStream stream = new NetworkStream(client);
```

自分の IP を取得

```
using System.Net;


public static IPAddress[] getAddress()
{
    return Dns.GetHostByName(Dns.GetHostName()).AddressList;
}
```

よーわからんが、これで使える IP アドレスの配列が取れるっぽい。

```
public static IPAddress[] getAddress()
```

```
{
    return Dns.Resolve(Dns.GetHostName()).AddressList;
}
```

こんな感じでもアリらしい。
むしろ、こっちの方がいいとか何とか。

--- 追記

.NET Framework 2.0 だと、こんな感じのコードで、IP アドレスの一覧が取得できるっぽい。

```
IPAddress[] address = Dns.GetHostAddresses(Dns.GetHostName());
```

ソケットのタイムアウト

```
// 接続待ち
Socket client = tcplistener.AcceptSocket();

// 2 秒受信しなかったらタイムアウトとして設定
client.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, 2000);
```

として、タイムアウトした場合

System.IO.IOException: 転送接続からデータを読み取れません。 ---> System.Net.Sockets.SocketException: 接続済みの呼び出し先が一定の時間を過ぎても正しく応答しなかったため、接続できませんでした。または接続済みのホストが応答しなかったため、確立された接続は失敗しました。

といった内容のえくせぷしょんが投げられる。

ちなみに、TcpClient ならもっと簡単にタイムアウトの設定ができるっぽい？

```
TcpClient client = new TcpClient("127.0.0.1", 12345);
client.ReceiveTimeout = 2000;
client.SendTimeout = 2000;
```

タイムアウトってのは、一定時間、応答が無かったり、通信しなかったときにエラーとして接続を切っちゃったりする機能。