МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И

ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**Отчёт**

**по лабораторной работе №2**

**по дисциплине**

**«Сетевые технологии»**

Выполнил: студент гр. пм-51 Бакулин А.В. \_\_\_\_\_\_\_\_\_\_\_

Проверил: преподаватель Прозорова Т.Г. \_\_\_\_\_\_\_\_\_\_\_

Киров 2014

**Задание 1.**

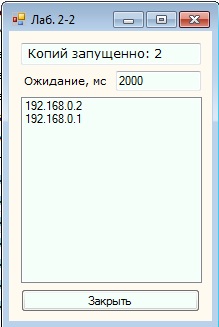
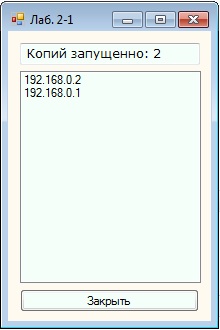
Разработать приложение, подсчитывающее количество копий себя, запущенных в локальной сети. Приложение должно использовать набор сообщений, чтобы информировать другие приложения о своѐм состоянии. После запуска приложение должно рассылать широковещательное сообщение о том что оно было запущено. Получив сообщение о запуске другого приложения оно должно сообщать этому приложению о том что оно работает. Перед завершением работы приложение должно информировать все известные приложения о том что оно завершает работу. На экран должен выводиться список IP адресов компьютеров на которых приложение запущено.

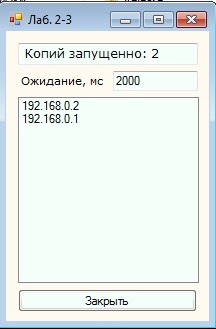
**Задание 2.**

Разработать приложение, подсчитывающее количество копий себя, запущенных в локальной сети. Приложение должно использовать рассылку широковещательных сообщений чтобы информировать другие приложение о том что оно запущено. Приложение считает другое приложение запущенным если в течении промежутка времени, равного нескольким интервалам между рассылками широковещательных сообщений, от него пришло сообщение.

**Задание 3.**

Разработать приложение, подсчитывающее количество копий себя, запущенных в локальной сети. Приложение должно использовать два подхода описанных в заданиях 1 и 2 вместе. Причѐм интервал между рассылками широковещательных сообщений должен быть больше чем в задаче 2.

****

****

**Задание 1**

namespace task2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private byte[] StartMsg, EndMsg;

private Thread listener;

private static Socket \_socket;

private static EndPoint \_point;

private static byte[] temp;

private static int delay;

private void Form1\_Load(object sender, EventArgs e)

{

StartMsg = Encoding.ASCII.GetBytes("Hello");

EndMsg = Encoding.ASCII.GetBytes("Bye");

temp = new Byte[10];

IPAddress myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[0];

listBox1.Items.Add(myip);

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

socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

socket.Bind(new IPEndPoint(myip, 11000));

socket.SendTo(StartMsg, new IPEndPoint(IPAddress.Broadcast, 11000));

socket.Close();

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

\_socket.ReceiveTimeout = 500;

IPEndPoint endp = new IPEndPoint(IPAddress.Any, 11000);

\_socket.Bind(endp);

\_point = (EndPoint)endp;

listener = new Thread(new ThreadStart(Listener));

listener.Start();

}

private void SetIP(IPAddress ip)

{

if (listBox1.InvokeRequired) listBox1.Invoke(new Action<IPAddress>((i) => listBox1.Items.Add(i)), ip);

else listBox1.Items.Add(ip);

}

private void RemoveIP(IPAddress ip)

{

if (listBox1.InvokeRequired) listBox1.Invoke(new Action<IPAddress>((i) => listBox1.Items.Remove(i)), ip);

else listBox1.Items.Remove(ip);

}

private void Listener()

{

while (true)

{

int length;

try

{

length = \_socket.ReceiveFrom(temp, ref \_point);

}

catch (Exception)

{

continue;

}

string pointstr = \_point.ToString();

IPAddress ip = IPAddress.Parse(pointstr.Substring(0, pointstr.IndexOf(':')));

if (length == 3) RemoveIP(ip);

else

{

SetIP(ip);

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

socket.SendTo(StartMsg, \_point);

socket.Close();

}

textBox1.Text = " Копий запущенно: " + listBox1.Items.Count;

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

listener.Abort();

\_socket.Close();

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

foreach (IPAddress ip in listBox1.Items) \_socket.SendTo(EndMsg, new IPEndPoint(ip, 11000));

\_socket.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

**Задание 2**

namespace task22

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private byte[] StartMsg, EndMsg;

private Thread listener;

private Socket \_socket;

private IPEndPoint \_point;

private IPAddress myip;

private List<IPAddress> list;

private static UdpClient udpclient;

private static Dictionary<IPAddress, int> data;

private void Form1\_Load(object sender, EventArgs e)

{

StartMsg = Encoding.ASCII.GetBytes("Hello");

EndMsg = Encoding.ASCII.GetBytes("Bye");

data = new Dictionary<IPAddress, int>();

list = new List<IPAddress>();

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[0];

listBox1.Items.Add(myip);

udpclient = new UdpClient(11000);

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

\_socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

\_socket.Bind(new IPEndPoint(myip, 11000));

\_point = new IPEndPoint(IPAddress.Broadcast, 11000);

timer1.Interval = Convert.ToInt32(textBox2.Text);

listener = new Thread(new ThreadStart(Listener));

listener.Start();

timer1.Start();

}

private void SetIP(IPAddress ip)

{

if (listBox1.InvokeRequired) listBox1.Invoke(new Action<IPAddress>((i) => listBox1.Items.Add(i)), ip);

else listBox1.Items.Add(ip);

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 11000);

byte[] temp = new byte[10];

string pstr;

IPAddress ip;

while (true)

{

try

{

temp = udpclient.Receive(ref point);

pstr = point.Address.ToString();

if (pstr == myip.ToString()) continue;

ip = IPAddress.Parse(pstr);

if (data.ContainsKey(ip)) data[ip] = 3;

else

{

data.Add(ip, 3);

SetIP(ip);

}

}

catch (Exception)

{

continue;

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

\_socket.SendTo(StartMsg, \_point);

for (int i = 1; i < listBox1.Items.Count; i++)

{

IPAddress tip = IPAddress.Parse(listBox1.Items[i].ToString());

if (--data[tip] == 0)

{

data.Remove(tip);

list.Add(tip);

}

}

foreach (IPAddress i in list) listBox1.Items.Remove(i);

list.Clear();

textBox1.Text = " Копий запущенно: " + listBox1.Items.Count;

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

listener.Abort();

\_socket.Close();

udpclient.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

timer1.Interval = Convert.ToInt32(textBox2.Text);

}

}

**Задание 3**

namespace task23

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private byte[] StartMsg, EndMsg;

private Thread listener;

private Socket \_socket;

private IPEndPoint \_point;

private IPAddress myip;

private List<IPAddress> list;

private static UdpClient udpclient;

private static Dictionary<IPAddress, int> data;

private void Form1\_Load(object sender, EventArgs e)

{

StartMsg = Encoding.ASCII.GetBytes("Hello");

EndMsg = Encoding.ASCII.GetBytes("Bye");

data = new Dictionary<IPAddress, int>();

list = new List<IPAddress>();

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[1];

listBox1.Items.Add(myip);

udpclient = new UdpClient(11000);

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

\_socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

\_socket.Bind(new IPEndPoint(myip, 11000));

\_point = new IPEndPoint(IPAddress.Broadcast, 11000);

\_socket.SendTo(StartMsg, \_point);

timer1.Interval = Convert.ToInt32(textBox2.Text);

listener = new Thread(new ThreadStart(Listener));

listener.Start();

timer1.Start();

}

private void SetIP(IPAddress ip)

{

if (listBox1.InvokeRequired) listBox1.Invoke(new Action<IPAddress>((i) => listBox1.Items.Add(i)), ip);

else listBox1.Items.Add(ip);

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 11000);

Byte[] temp;

string pstr, recstr;

IPAddress ip;

while (true)

{

try

{

temp = udpclient.Receive(ref point);

recstr = Encoding.ASCII.GetString(temp);

pstr = point.Address.ToString();

if (pstr == myip.ToString()) continue;

ip = IPAddress.Parse(pstr);

if (recstr == "Bye") data[ip] = -1;

else

if (data.ContainsKey(ip)) data[ip] = 6;

else

{

data.Add(ip, 6);

SetIP(ip);

}

}

catch (Exception)

{

continue;

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

\_socket.SendTo(StartMsg, \_point);

for (int i = 1; i < listBox1.Items.Count; i++)

{

IPAddress tip = IPAddress.Parse(listBox1.Items[i].ToString());

if (--data[tip] <= 0)

{

data.Remove(tip);

list.Add(tip);

}

}

foreach (IPAddress i in list) listBox1.Items.Remove(i);

list.Clear();

textBox1.Text = " Копий запущенно: " + listBox1.Items.Count;

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

listener.Abort();

udpclient.Close();

foreach (IPAddress ip in listBox1.Items) \_socket.SendTo(EndMsg, new IPEndPoint(ip, 11000));

\_socket.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

timer1.Interval = Convert.ToInt32(textBox2.Text);

}

**Вывод**

В результате выполнения лабораторной работы исследованы различные способы взаимодействия приложений внутри сети. Созданная программа взаимодействует со своими копиями двумя различными способами, а также комбинацией этих способов. Данная технология может быть использована при наличии ограничения на количество лицензионных копий приложений.