

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HÙNG YÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP THỰC HÀNH

LẬP TRÌNH MẠNG MÁY TÍNH

BIÊN SOẠN: BM CÔNG NGHỆ PHẦN MỀM

TRÌNH ĐỘ: ĐẠI HỌC

Biên soạn: Nguyễn Minh Quý

HÙNG YÊN THÁNG 9/2012

BÀI THỰC HÀNH SỐ 1: Sử dụng lớp UdpClient

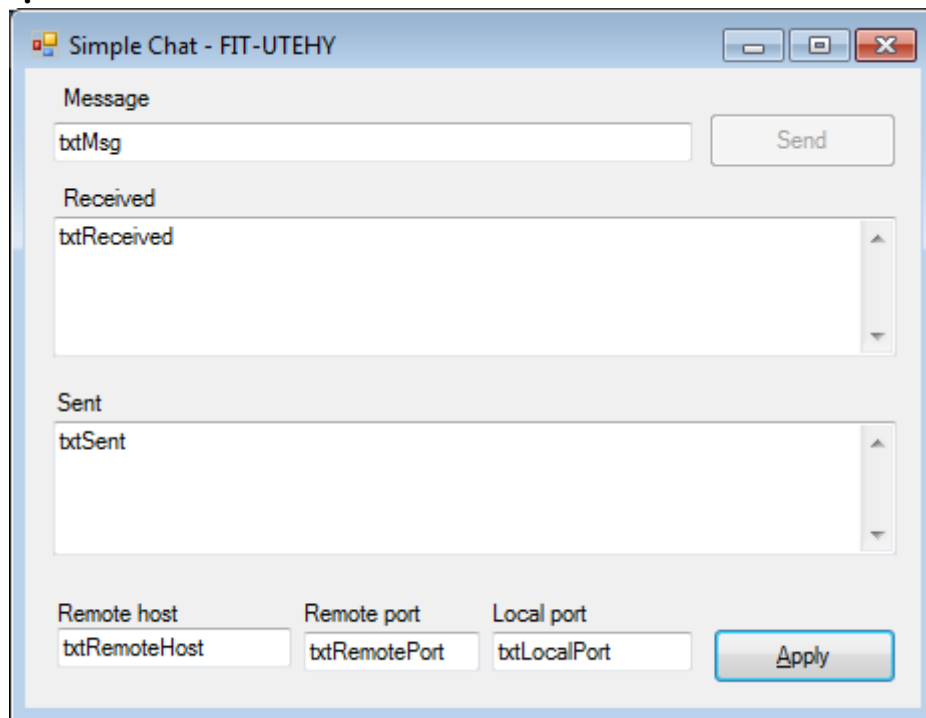
MỤC TIÊU: Kết thúc bài thực hành này, sinh viên có thể

- Sử dụng được lớp UdpClient để xây dựng ứng dụng Chat đơn giản giữa hai máy tính
- Vận dụng để xây dựng các ứng dụng thực tế khác trên cơ sở giao thức Udp

NỘI DUNG THỰC HÀNH

Xây dựng chương trình Chat đơn giản giữa hai máy tính với yêu cầu sau đây:

1. Giao diện



Giao diện chương trình (kèm tên các điều khiển)

2. Đặc tả xử lý

- Khi click nút Send (btnSend) thì sẽ gửi xâu trong ô textbox txtMsg tới ứng dụng nhận có địa chỉ và cổng tương ứng trong txtRemoteHost và txtRemotePort. Xâu gửi đi cũng đồng thời được chèn vào txtSent.
- Khi có dữ liệu gửi đến thì ứng dụng sẽ nhận và chèn vào txtReceived.

3. Hướng dẫn

- Khai báo biến toàn cục UdpClient để sử dụng trong cả gửi và nhận dữ liệu
- Sử dụng điều khiển Timer hoặc vòng lặp để thường xuyên kiểm tra dữ liệu gửi đến. Dùng timer sẽ dễ dàng hơn.
- Cần kiểm tra có dữ liệu trước khi gọi hàm Receive để tránh bị tình trạng “Blocking”

4. Chương trình mẫu

```
using System;  
using System.Text;  
using System.Windows.Forms;
```

```

using System.Net;
using System.Net.Sockets;
//Chat – Simple: By Software Engineering Department - FIT-UTEHY 2012.
namespace Chat_Simple
{
    public partial class Form1 : Form
    {
        UdpClient udp;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Tạm thời chưa cho hoạt động khi chưa đặt thông số
            btnSend.Enabled = false;
            timer1.Enabled = false;
        }

        //Thiết lập thông số cổng theo giá trị do người dùng nhập
        private void btnApply_Click(object sender, EventArgs e)
        {
            //Do phải thiết lập theo địa chỉ và cổng mới nhất nhập vào
            //nên cần phải xóa (close) udp cũ đã xin cấp phát trước đó
            if (udp != null) udp.Close();

            //Tạo udp theo các thông số cổng được nhập trong localport
            udp = new UdpClient(Int32.Parse(txtLocalPort.Text));

            //Bắt đầu cho timer hoạt động và cho phép gửi dữ liệu khi có
            //thông số nhập vào trong remotehost, remoteport, localport
            timer1.Enabled = true;
            timer1.Interval = 100; //100 miligiay = 1/10 giây
            btnSend.Enabled = true;
        }

        //Gửi dữ liệu sang ứng dụng khác
        private void btnSend_Click(object sender, EventArgs e)
        {
            byte[] Data = Encoding.UTF8.GetBytes(txtMsg.Text);
            udp.Send(Data, Data.Length, txtRemoteHost.Text,
                    Int32.Parse(txtRemotePort.Text));

            //chèn nội dung vừa gửi vào trong textbox sent
            txtSent.Text = txtMsg.Text + "\r\n" + txtSent.Text ;
        }

        //Nhận dữ liệu từ ứng dụng khác
        private void timer1_Tick(object sender, EventArgs e)

```

```

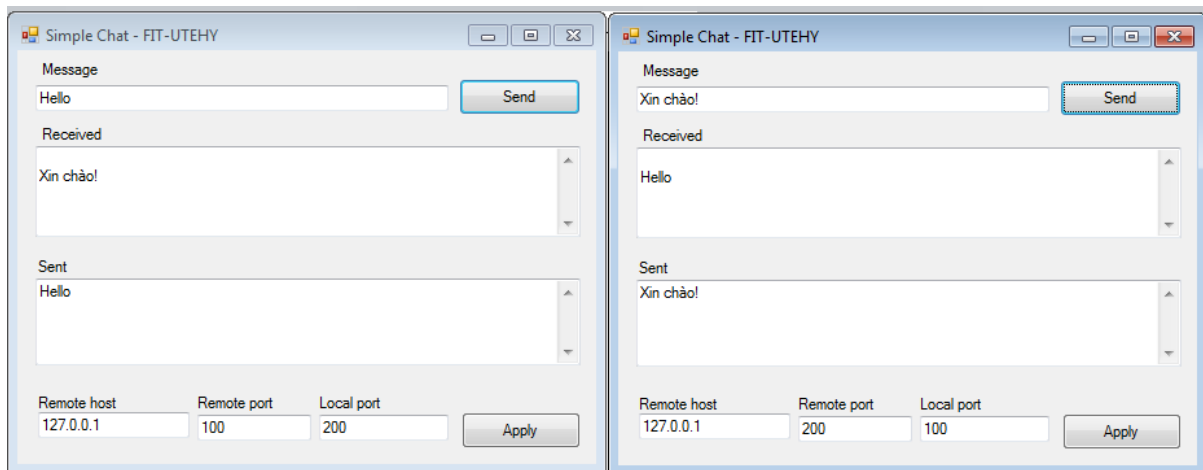
{
    //nếu có dữ liệu gửi đến
    if (udp.Available > 0)
    {
        byte[] Data;
        IPEndPoint iep = new IPEndPoint(
            IPAddress.Parse(txtRemoteHost.Text),
            Int32.Parse(txtRemotePort.Text));

        Data = udp.Receive(ref iep);
        txtReceived.Text += "\r\n" + Encoding.UTF8.GetString(Data);
    }
}
}
}

```

5. Chạy chương trình

Biên dịch chương trình, sau đó chạy file exe 2 lần. Nhập giá trị như hình sau:



*** Sau khi nhập (lưu ý: giá trị cổng chéo nhau giữa hai ứng dụng), click nút Apply.

6. Nhận xét

- Không dùng vòng lặp để kiểm tra dữ liệu gửi đến vì Timer tương tự như vòng lặp.
- Có thể tạo một tuyến (thread) trong đó có vòng lặp để thực hiện việc nhận dữ liệu thay vì timer.
- Để xuống dòng trong textbox, cần thêm chuỗi ký tự “\r\n”
- Trong phần khai báo, ta viết: `UdpClient udp`; thay vì viết như thường lệ là: `UdpClient udp = new UdpClient(giá trị cổng)`; Lý do là vì số hiệu cổng sẽ được lấy từ ô textbox `txtLocalPort` do người dùng nhập vào khi chạy chương trình.

7. Bài tập làm thêm

Mở rộng chương trình trên để ngoài chức năng chat có thể từ một máy ra lệnh cho máy kia tắt, khởi động lại máy tính.

Gợi ý: Bên gửi chỉ cần nhập nội dung gửi đi theo dạng: #01 , hoặc #02 hoặc một xâu thông thường. Còn đối với phía nhận thì sẽ kiểm tra 3 ký tự đầu tiên của xâu nhận về (Sử dụng hàm SubString(0, 3) để trích 3 ký tự đầu tiên). Nếu là “#01” thì khởi động lại máy; nếu là “#02” thì tắt máy, trái lại thì hiểu đơn giản đó là dòng chat thông thường (khi đó sẽ đưa vào ô txtReceived như bài thực hành).

- Dòng lệnh để thực hiện khởi động lại máy:
System.Diagnostics.Process.Start(“Shutdown”, “-r -t 3”);
- Dòng lệnh để thực hiện tắt máy:
System.Diagnostics.Process.Start(“Shutdown”, “-s -t 3”);

BÀI THỰC HÀNH SỐ 2: Gửi tin quảng bá

Mục tiêu: Kết thúc bài thực hành này, sinh viên có thể:

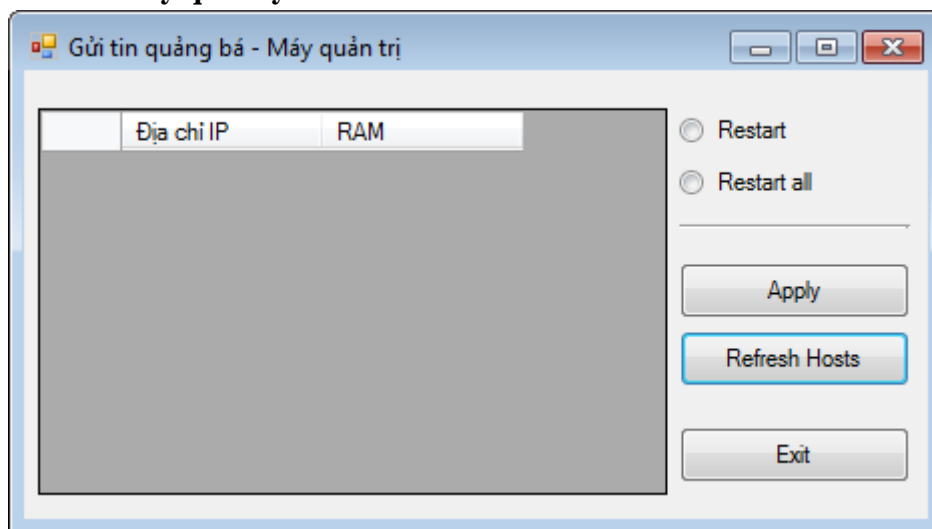
- Thực hiện gửi tin quảng bá tới nhiều máy tính đồng thời trong mạng
- Vận dụng xây dựng các ứng dụng như Hỗ trợ quản lý phòng mạng, giám sát các máy tính trong phòng thực hành...

NỘI DUNG

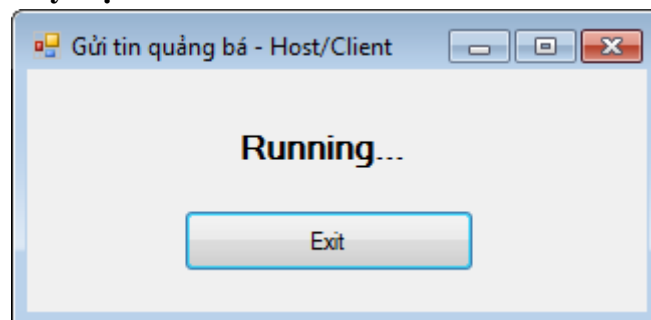
Xây dựng chương trình có yêu cầu như sau:

1. Về giao diện

a. Trên máy quản lý



b. Trên các máy trạm



2. Yêu cầu

a. Trên máy quản lý

- Khi người dùng click nút “**Refresh hosts**” thì danh sách địa chỉ (tên máy) của các máy khác trong mạng sẽ liệt kê trong danh sách (GridView)
- Khi người dùng chọn một địa chỉ trong gridview và chọn “Restart” sau đó click Apply thì tiến hành restart máy ứng với địa chỉ tương ứng.
- Khi người dùng chọn Restart All và click Apply thì restart lại tất cả các máy có địa chỉ trong gridview.

b. Trên máy trạm

- Máy trạm khi khởi động sẽ chạy ngầm và lắng nghe các lệnh gửi đến từ máy quản trị. Nếu có yêu cầu restart thì restart lại máy, nếu có yêu cầu gửi địa chỉ máy mà nó đang chạy trên đó thì tiến hành gửi.

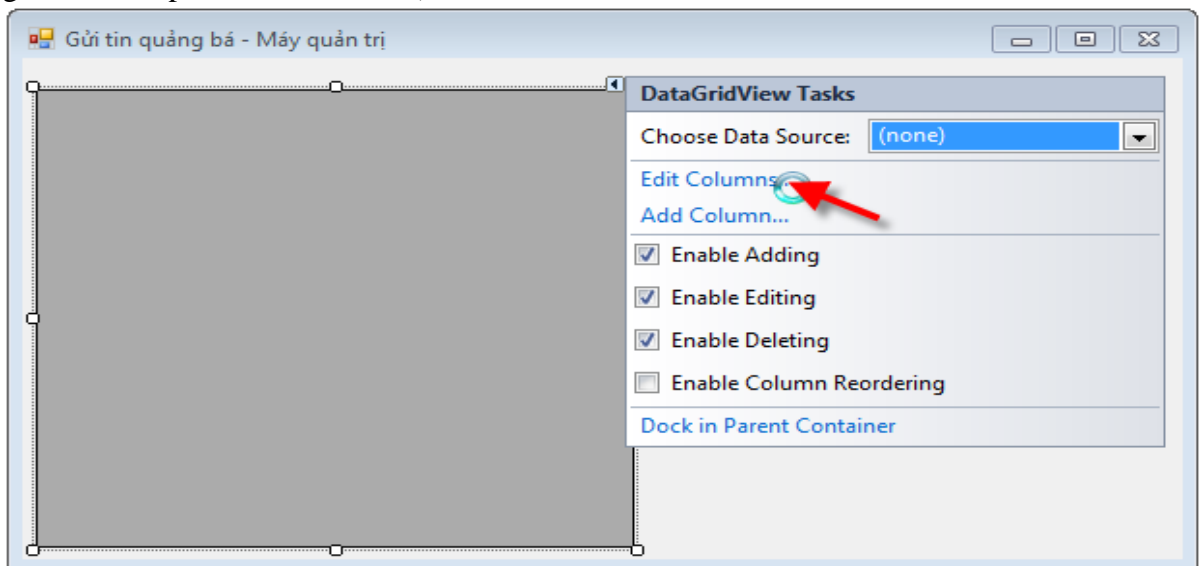
3. Hướng dẫn

- Để các máy trạm gửi tên máy của nó tới ứng dụng quản lý, ta tiến hành gửi một gói tin quảng bá tới máy trạm với quy ước: “please_send_me_ip” để yêu cầu gửi địa chỉ hoặc tên máy. Sau đó sẽ dùng một timer để liên tục kiểm tra dữ liệu trả về từ các máy trạm. Việc lấy tên máy trên các máy trạm có thể dùng hàm Dns.GetHostName(). Hàm này trả về tên máy mà nó đang chạy.
- Để restart một máy, ta chỉ việc lấy tên của máy mà người dùng chọn trên gridview, sau đó gửi gói tin là “please_restart”. Máy trạm khi nhận được gói tin này sẽ thực hiện restart máy bằng hàm System.Diagnostics.Process.Start(“Shutdown”, “-r -t 3”). Lưu ý, trong hàm gửi dữ liệu đi, ta có thể viết tên máy thay vì địa chỉ IP: Send(Data, length, **HostName**, port).
- Để Restart tất cả các máy, ta chỉ đơn giản gửi cho mọi máy gói tin “please_restart” bằng cơ chế gửi quảng bá.
- Để truy cập đến một ô tại hàng i, cột j trong gridview có tên là dgvHost, theo cú pháp:

dgvHost.Rows[i].Cells[j].Value

4. Thực hiện

Bước 1: Tạo ứng dụng winform và đưa điều khiển gridview vào form để lưu danh sách địa chỉ IP các máy. Sau đó click chọn Edit columns (bấm vào biểu tượng mũi tên góc trên bên phải của điều khiển)



Bước 2: Thêm một cột “Địa chỉ IP”.

Bước 3: Thêm các điều khiển vào form

STT	Điều khiển	Tên	Nhân
1.	Raido button	optRestart	Restart
2.	Raido button	optRestartAll	Restart all
3.	Button	btnRefresh	Refresh Hosts
4.	Button	btnApply	Apply
5.	Button	btnExit	Exit applicaceion
6.	Timer	Timer1	-

Bước 4: Viết code

Code cho ứng dụng phía máy quản trị

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace TH_so2___Gui_tin_quang_ba___Admin
{
    public partial class Form1 : Form
    {
        //Udp trên máy người quản trị sẽ sử dụng (bind) cổng 100
        UdpClient udp = new UdpClient(100);

        public Form1()
    }
}

```



```

{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
    timer1.Enabled = true;
    timer1.Interval = 100;
    dgrHosts.AllowUserToAddRows = false; //Không cho phép tự động thêm dòng mới
    dgrHosts.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
}

//khi refresh, chỉ việc xóa danh sách trong gridview
//và gửi lệnh yêu cầu (broadcast) các máy gửi lại địa chỉ IP
private void btnRefresh_Click(object sender, EventArgs e)
{
    //xóa nội dung trong gridview
    dgrHosts.Rows.Clear();

    //Gửi tín hiệu để yêu cầu các máy trạm (cổng 200) gửi lại IP
    byte[] Data = Encoding.UTF8.GetBytes("please_send_me_your_ip");
    udp.EnableBroadcast = true;

    //Gửi thông báo tới tất cả các ứng dụng đang sử dụng giao thức UDP, cổng 200
    udp.Send(Data, Data.Length, "255.255.255.255", 200);
}

//Timer này sẽ nhận các thông tin gửi từ các máy trạm (gồm IP)
private void timer1_Tick(object sender, EventArgs e)
{
    if (udp.Available > 0)
    {
        //Khai báo một IPEndPoint cho phép nhận từ nhiều máy gửi tới
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 200);
        udp.EnableBroadcast = true;
        byte[] Data = udp.Receive(ref iep);
        String HostAddress = Encoding.UTF8.GetString(Data);

        //thêm một dòng mới vào gridview để lưu địa chỉ IP vừa gửi tới.
        dgrHosts.Rows.Add();

        int Dòng_Cuối = dgrHosts.Rows.Count - 1;
        dgrHosts.Rows[Dòng_Cuối].Cells[0].Value = HostAddress;
    }
}

//xử lý theo các tùy chọn của người dùng (Restart, restartall)
private void btnApply_Click(object sender, EventArgs e)
{
    //chỉ gửi tới một IP được chọn

```

```

if (optRestart.Checked == true)
{
    //nếu không chọn máy nào thì thoát.
    if (dgrHosts.SelectedRows.Count == 0) return;

    //Chỉ gửi tín hiệu restart đến địa chỉ IP được chọn
    String HostAddress = dgrHosts.SelectedRows[0].Cells[0].Value.ToString(); //lấy IP
    byte[] Data = Encoding.UTF8.GetBytes("please_restart");
    udp.Send(Data, Data.Length, HostAddress, 200);
}

//Gửi tới tất cả các máy đang sử dụng cổng 200, UDP
if (optRestartAll.Checked == true)
{
    byte[] Data = Encoding.UTF8.GetBytes("please_restart");
    udp.EnableBroadcast = true;
    udp.Send(Data, Data.Length, "255.255.255.255", 200); //địa chỉ quảng bá
}

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}
}

```

Code cho ứng dụng chạy trên máy trạm

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;

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

namespace TH_so_2___Gui_tin_quang_ba___Host
{
    public partial class Form1 : Form
    {
        //Tạo một thẻ hiện UdpClient, sử dụng cổng 200
        UdpClient udp = new UdpClient(200);
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

```

private void Form1_Load(object sender, EventArgs e)
{
    timer1.Enabled = true;
    timer1.Interval = 100;
}

//nếu có dữ liệu gửi đến từ máy quản trị thì xử lý và gửi lại
private void timer1_Tick(object sender, EventArgs e)
{
    if (udp.Available > 0)
    {
        //vì bên gửi gửi dữ liệu kiểu broadcast (quảng bá) từ cổng 100
        //nên địa chỉ nhận đặt là IPAddress.Any, cổng là 100
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 100);
        udp.EnableBroadcast = true;

        byte [] Data = udp.Receive(ref iep);    //nhận dữ liệu
        String S = Encoding.UTF8.GetString(Data); //chuyển thành xâu để xử lý

        if (S.CompareTo("please_send_me_your_ip") == 0)
        {
            String Hostname = Dns.GetHostName();
            Data = Encoding.UTF8.GetBytes(Hostname);
            udp.Send(Data, Data.Length, iep);
        }

        if (S.CompareTo("please_restart") == 0)
        {
            System.Diagnostics.Process.Start("shutdown", "-r -t 3");
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}

```

5. Nhận xét

- Khi nhận dữ liệu gửi tới từ một máy, ta thường viết `udp.Receive(ref iep)`, thì **iep** này sẽ chứa thông tin (địa chỉ, cổng) của máy gửi. Vì vậy, khi cần reply lại thì ta sử dụng chính giá trị này khi gửi, lúc đó sử dụng hàm `Send` có dạng: `udp.Send(Data, Data.Length, iep)` thay vì sử dụng hàm `Send` dạng: `Send(Data, Data.Length, Host, Port)`.
- Có thể lấy tên của máy tính bằng cách gọi hàm: `Dns.GetHostName()`
- Có thể thay timer bằng cơ chế đa luồng (đa tuyến – Multi threading) khi nhận dữ liệu gửi đến. Ví dụ có thể viết lại chương trình cho phía máy trạm cụ thể như sau:

*** Nhớ thêm khai báo sau trong chương trình: **using** System.Threading;

Chương trình phía máy trạm (Sử dụng cơ chế Threading)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;

using System.Threading;
using System.Net;
using System.Net.Sockets;

namespace TH_so_2___Gui_tin_quang_ba___Host
{
    public partial class Form1 : Form
    {
        //Tạo một thể hiện UdpClient, sử dụng cổng 200
        UdpClient udp = new UdpClient(200);
        bool Stop = false;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Tạo một tuyến để chạy riêng hàm Xử_Lý_Dữ_Liệu_Gửi_Đến
            Thread th = new Thread(new ThreadStart(Xử_Lý_Dữ_Liệu_Gửi_Đến));
            th.Start();
        }

        void Xử_Lý_Dữ_Liệu_Gửi_Đến()
        {
            while (Stop==false)
            {
                if (udp.Available > 0)
                {
                    //vì bên gửi gửi dữ liệu kiểu broadcast (quảng bá) từ cổng 100
                    //nên địa chỉ nhận đặt là IPAddress.Any, cổng là 100
                    IPEndPoint iep = new IPEndPoint(IPAddress.Any, 100);
                    udp.EnableBroadcast = true;

                    byte[] Data = udp.Receive(ref iep);    //nhận dữ liệu
                    String S = Encoding.UTF8.GetString(Data); //chuyển thành chuỗi để xử lý

                    if (S.CompareTo("please_send_me_ip") == 0)
                    {

```

```

        String Hostname = Dns.GetHostName();
        Data = Encoding.UTF8.GetBytes(Hostname);
        udp.Send(Data, Data.Length, iep);
    }

    if (S.CompareTo("please_restart") == 0)
    {
        System.Diagnostics.Process.Start("shutdown", "-r -t 3");
    }
}

//Báo cho hệ điều hành biết là dành bớt thời gian xử lý ứng dụng khác
Application.DoEvents();
}

//Kết thúc tuyến thì thoát khỏi ứng dụng
Application.Exit();
}
//nếu có dữ liệu gửi đến từ máy quản trị thì xử lý và gửi lại

private void btnExit_Click(object sender, EventArgs e)
{
    //người dùng đã click nút Exit, đặt biến Stop = true để kết thúc vòng lặp trong thread
    Stop = true;
}
}
}

```

Ghi chú:

- Muốn cho một chương trình con (Tuyến) chạy song song với chương trình chính thì ta chỉ việc viết một hàm (không tham số) sau đó truyền cho Thread. Sau đó chạy hàm này bằng cách gọi phương thức Start của Thread vừa tạo.
- Một thread khi chạy nó có thể vẫn tồn tại ngay cả chương trình chính đã thoát. Vì vậy, khi thoát khỏi chương trình chính thì cũng cần phải tắt thread đã tạo ra. Ở đây, vì trong Thread có một vòng lặp vô hạn, do vậy khi kết thúc chương trình chính, ta chỉ việc đặt một biến nhớ là Stop = True và đưa điều kiện này vào trong vòng lặp của Thread.
- Có thể sử dụng cơ chế đa tuyến này cho phần nhận dữ liệu bên phía máy quản trị.

6. Bài tập mở rộng

Mở rộng bài toán trên với yêu cầu sau: Phía máy trạm sẽ gửi tên máy và Dung lượng RAM của máy tính. Hướng dẫn: Máy trạm sẽ gửi nội dung dạng <Tên_Máy>\$<Dung lượng RAM>. Phía máy quản lý sẽ tách riêng 2 phần này ra bằng hàm Split.

Để lấy dung lượng RAM của máy, tìm trên Google với từ khóa: **C# get RAM**

BÀI THỰC HÀNH SỐ 3

SỬ DỤNG TcpClient ĐỂ XÂY DỰNG ỨNG DỤNG CLIENT

MỤC TIÊU: Kết thúc bài thực hành, sinh viên có thể

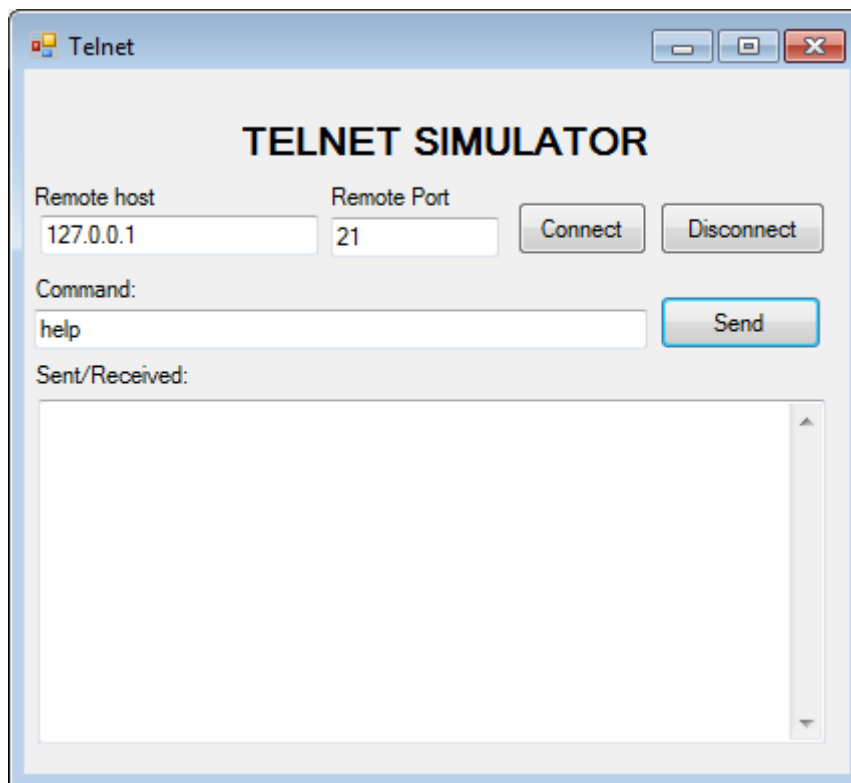
- Sử dụng được lớp TcpClient để xây dựng ứng dụng giả lập Telnet
- Vận dụng để xây dựng các ứng dụng Client thực tế.

NỘI DUNG:

1. Yêu cầu

Xây dựng chương trình tương tự như Telnet (Một chương trình cho phép người dùng kết nối và gửi/nhận dữ liệu với server. Trên windows có thể chạy chương trình telnet bằng cách vào chế độ Command và gõ lệnh: Telnet).

Giao diện như sau:



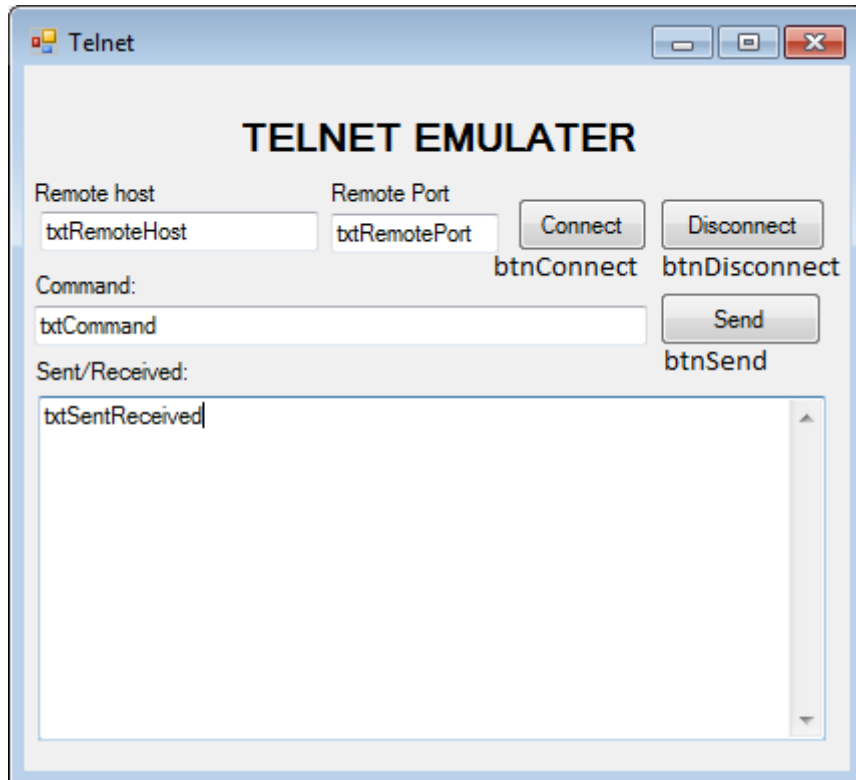
Giao diện chương trình

Đặc tả xử lý:

- Khi chương trình khởi động, nút Disconnect và Send ở trạng thái cấm (Disable).
- Click nút Connect thì thực hiện kết nối đến server có địa chỉ và cổng tương ứng như trong textbox remotesite và remoteport. Nếu kết nối thành công thì đặt nút Disconnect, Send ở trạng thái Enabled và nút Connect ở trạng thái Disabled.
- Khi click Send thì gửi nội dung trong ô Command.
- Nội dung gửi đi hoặc nhận về được lưu trong textbox Sent/Received.

2. Hướng dẫn

Đặt tên các điều khiển như sau:



Đặt tên cho các điều khiển trên form

3. Minh họa chương trình mẫu

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace TK8LC_Telnet_Simulator
{
    public partial class Form1 : Form
    {
        TcpClient client;
        Thread Th;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Bỏ qua kiểm tra xung đột giữa các luồng khi truy cập vào điều khiển trên form
        }
    }
}
```

```

        System.Windows.Forms.Control.CheckForIllegalCrossThreadCalls = false;
    }

    private void btnConnect_Click(object sender, EventArgs e)
    {
        client = new TcpClient();
        client.Connect(txtRemoteHost.Text, Int32.Parse(txtRemotePort.Text));
        Th = new Thread(new ThreadStart(DocDuLieu));
        Th.Start();
    }

    //Tuyến (thread) đọc dữ liệu gửi đến từ Server
    void DocDuLieu()
    {
        while (true)
        {
            Thread.Sleep(10) ;           //Dừng 10ms để đợi dữ liệu đến hết vào buffer
            StreamReader sr = new StreamReader(client.GetStream());
            string S = sr.ReadLine();
            txtRcv.Text += S + "\r\n";

        }
    }

    private void btnSend_Click(object sender, EventArgs e)
    {
        StreamWriter sw = new StreamWriter(client.GetStream());
        sw.WriteLine(txtMsg.Text);
        sw.Flush();
    }

    private void btnDisconnect_Click(object sender, EventArgs e)
    {
        Th.Abort();           //Dừng tuyến đang chạy
        client.Close();       //Đóng (Giải phóng) TcpClient
    }
}

```

4. Chạy chương trình

Để kiểm tra được chương trình, chúng ta cần chạy một ứng dụng Server. Ví dụ vào Google và search một server (FTP) chẳng hạn. Dưới đây là chỉ download một FTP Server đơn giản: <http://www.argosoft.com/rootpages/Download.aspx>

Sau khi cài đặt, chạy chương trình, nó sẽ mặc định nghe ở cổng 21. Ghi nhớ số hiệu cổng này (thuộc tính Remote Port) để kết nối về sau.

5. Nhận xét

- Khi đọc dữ liệu từ một stream, thì hàm ReadLine() của streamReader sẽ làm việc ở chế độ đồng bộ, nghĩa là nó sẽ chờ cho đến khi có dữ liệu gửi đến thì các câu lệnh đứng sau nó mới được thực hiện.

- Khi đóng Client thì phải đảm bảo rằng tuyến (thread) đang chạy phải kết thúc trước. Vì nếu ta đóng client khi một tuyến lại đang sử dụng đến Client đó sẽ báo lỗi. Do vậy, thứ tự câu lệnh sau đây là quan trọng:

```
Th.Abort();           //Dừng tuyến đang chạy
client.Close();       //Đóng (Giải phóng) TcpClient
```

- Phần dừng lại để đợi dữ liệu đến hết vào server rất quan trọng, vì nếu không có dòng lệnh này, những dòng gửi đến sau có thể không được StreamReader đọc hết (dẫn đến tình trạng không đọc hết được dữ liệu gửi đến)

6. Bài tập mở rộng

- Viết chương trình thực hiện “đăng nhập” vào Server trước khi tiến hành các thao tác gửi dữ liệu.
- Giả sử việc đăng nhập được tiến hành theo giao thức đơn giản như sau:
 - o Client: Kết nối đến Server
 - o Server: Gửi xâu trả về “200 Welcome..... Please give me username for authorization”
 - o Client: user <tên người dùng>
 - o Server: 200 OK, please give me the password (hoặc **500 User name not exist, please give available user!** Nếu user không tồn tại)
 - o Client: pass <mật khẩu>
 - o Server: 200 OK, you are logged in. Now you can send me some commands (hoặc **501 Password is not correct. Please resend the password**)
 - o Client: gửi các xâu như bình thường.
- Gợi ý: User name và password sẽ do người dùng nhập vào trong 2 textbox. Khi người dùng click vào nút “Login” trên form thì sẽ tiến hành kiểm tra. Mỗi khi kiểm tra, cần phải đọc dữ liệu trả về từ server. Trong dữ liệu trả về đó, trích phần mã (3 số đầu) để kiểm tra.

BÀI THỰC HÀNH SỐ 4

SỬ DỤNG TcpListener ĐỂ XÂY DỰNG ỨNG DỤNG SERVER

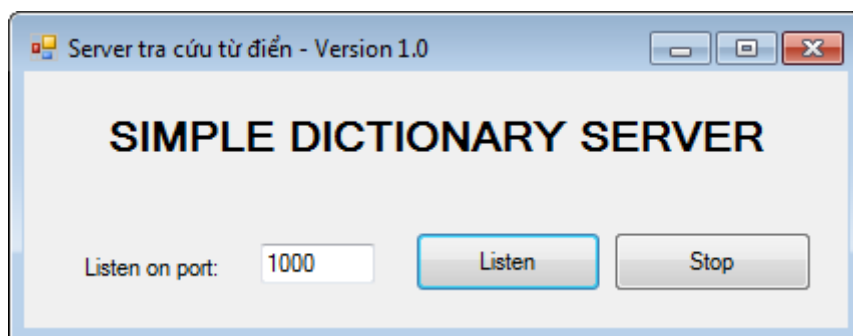
MỤC TIÊU: Kết thúc bài thực hành, sinh viên có thể

- Sử dụng được lớp TcpListener để xây dựng ứng dụng Tra cứu từ điển đơn giản
- Vận dụng để xây dựng các ứng dụng Server thực tế theo yêu cầu.

NỘI DUNG:

1. Yêu cầu

Viết chương trình TCP Server có giao diện như sau:

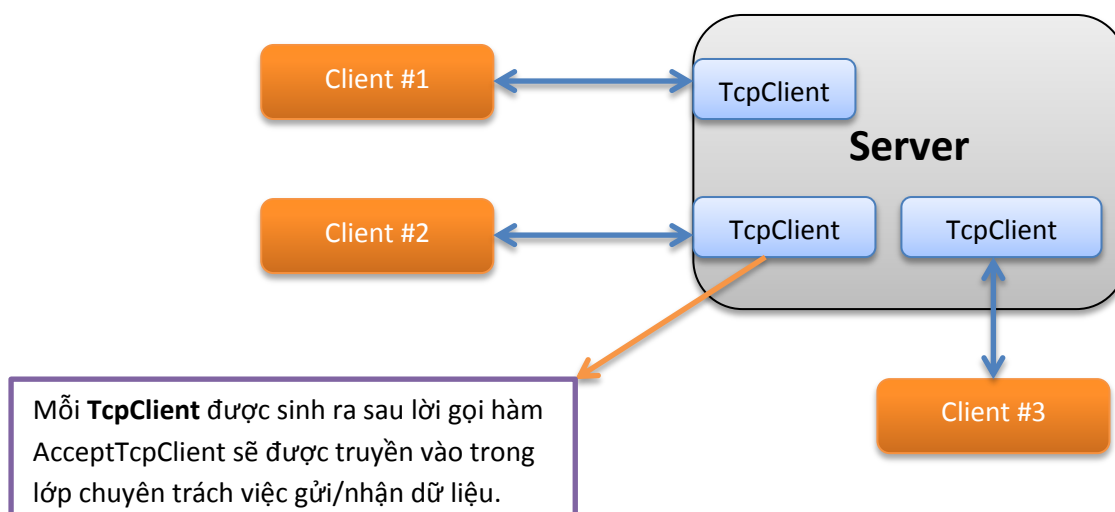


2. Đặc tả chức năng

- Khi click nút Listen, server lắng nghe trên cổng được nhập vào từ người dùng, mặc định là cổng 1000.
- Khi click nút Stop thì Server dừng lắng nghe
- Khi có yêu cầu gửi đến từ client (Nội dung là một từ Tiếng Anh), Chương trình sẽ thực hiện việc kiểm tra (tra cứu) và trả về nghĩa Tiếng việt tương ứng cho phía Client.
- Server phải có khả năng chấp nhận và xử lý nhiều kết nối đồng thời.

3. Hướng dẫn

Để chấp nhận nhiều kết nối đồng thời, ta định nghĩa một lớp chuyên để thực hiện việc gửi và nhận dữ liệu với Client. Lớp này sẽ nhận tham số đầu vào là một TcpClient (đối tượng trả về từ phương thức AcceptTcpClient). Trong lớp này sẽ có một phương thức dùng để nhận và xử lý dữ liệu và sẽ được chạy trong một Thread riêng.



4. Minh họa chương trình mẫu

```
using System;
using System.Text;
using System.Windows.Forms;

using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace TK9LC_TCPServer_V1
{
    public partial class Form1 : Form
    {
        TcpListener Server;
        bool Stop = false;

        public Form1()
        {
            InitializeComponent();
        }

        private void btnListen_Click(object sender, EventArgs e)
        {
            //Lắng nghe kết nối tại cổng txtLocalPort, trên tất cả các card mạng.
            Server = new TcpListener(IPAddress.Any, int.Parse(txtLocalPort.Text));
            Server.Start();

            //Đặt lại trạng thái các nút nhấn
            btnStop.Enabled = true;
            btnListen.Enabled = false;

            TcpClient Client;
            while (!Stop)
            {
                if (Server.Pending()) //Nếu có kết nối đến từ client
                {
                    //Chấp nhận khi có kết nối đến và lưu kết quả vào biến Client
                    Client = Server.AcceptTcpClient();

                    //Chuyển Client vừa được tạo ra này cho lớp chuyên xử lý gửi/nhận
                    XuLyClient objXuly = new XuLyClient(Client);

                    //Chạy phương thức XuLy của ObjXuLy vừa tạo trong 1 tuyến riêng
                    Thread Th = new Thread(new ThreadStart(objXuly.XuLy));
                    Th.Start();
                }
                Application.DoEvents();
            }
        }

        private void btnStop_Click(object sender, EventArgs e)
        {
            Stop = true;
        }
    }
}
```

```

        Thread.Sleep(100); //Chờ 100ms để cho vòng lặp (while (!Stop)) kết thúc.

        //Stop Server
        Server.Stop();

        //Đặt lại trạng thái các nút nhấn cho phù hợp
        btnStop.Enabled = false;
        btnListen.Enabled = true;
    }
}

//Lớp được thiết kế để Server ủy quyền xử lý mỗi kết nối được tạo ra.
class XuLyClient
{
    TcpClient Client;

    public XuLyClient(TcpClient client)
    {
        Client = client; //Lưu tham chiếu đến Client do server đã Accepted
    }

    //Phương thức này sẽ được chạy trong một tuyến riêng biệt
    public void XuLy()
    {
        while (true)
        {
            StreamReader sr = new StreamReader(Client.GetStream());
            StreamWriter sw = new StreamWriter(Client.GetStream());

            String English, VietNameese;
            English = sr.ReadLine();

            if (English.ToLower() == "hello") VietNameese = "Xin chào!";
            else if (English.ToLower() == "goodbye") VietNameese = "Tạm biệt!";
            else if (English.ToLower() == "computer") VietNameese = "Máy tính";
            else VietNameese = "Không có trong từ điển...";

            sw.AutoFlush = true;
            sw.WriteLine(VietNameese);
        }
    }
}

```

5. Nhận xét

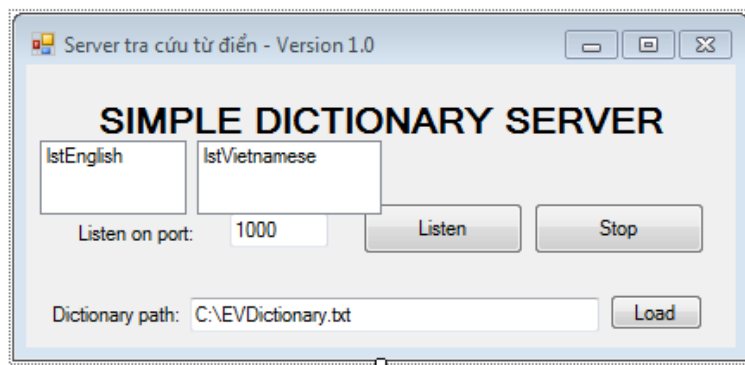
- Câu lệnh: `Server = new TcpListener(IPAddress.Any, int.Parse(txtLocalPort.Text));` trong đó địa chỉ `IPAddress.Any` để hàm ý rằng chương trình server sẽ lắng nghe trên cổng của tất cả các card mạng sẵn có trong máy tính (một máy tính có thể có nhiều card mạng).
- Ta thêm biến `Stop` vào trong vòng lặp để đảm bảo khi người dùng nhấn nút `Stop` thì ta chỉ việc đặt `Stop=true`, lúc đó vòng lặp sẽ kết thúc (Không phải lặp vô hạn).

- Ta phải viết một lớp chuyên xử lý dữ liệu gửi từ Client vì server có thể chấp nhận nhiều kết nối đồng thời. Nếu không tạo một lớp riêng thì chỉ có thể xử lý được một Client mà thôi.
- Ở bài tập này, chúng ta giả định từ điển chỉ có 3 từ để dễ hiểu được cơ chế xử lý của ứng dụng Server. Trên thực tế số lượng các từ trong từ điển có thể nhiều hơn và người ta có thể để trong tệp hoặc trong một bảng CSDL. Khi đó, thay vì các câu lệnh if, **else if** ở trên chúng ta có thể sử dụng các kỹ thuật truy vấn CSDL như thông thường (Giá trị dùng đặt điều kiện trong câu lệnh select là từ tiếng Anh).

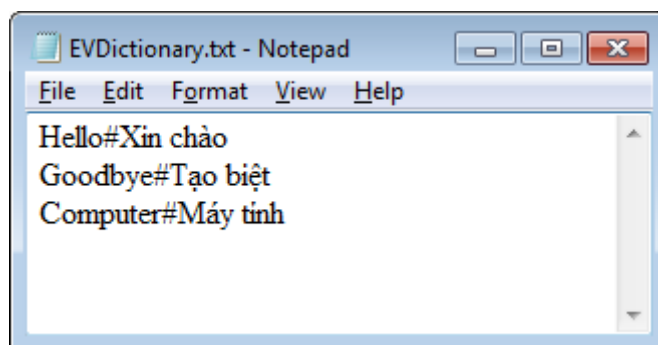
6. Bài tập mở rộng

6.1 Viết lại chương trình trên với yêu cầu như sau: Từ điển sẽ được lưu trong một cơ sở dữ liệu SQL, trong đó có một bảng với cấu trúc như sau: tblEV(**Id**, English, VietNameese). Khi có yêu cầu tra cứu từ Client thì server sẽ thay vì kiểm tra bằng câu lệnh If như ở trên thì sẽ tìm kiếm trong Database và trả về kết quả.

6.2 Hoặc viết lại chương trình trên với giao diện như sau:



Trong đó, nội dung từ điển sẽ lưu vào trong một tệp có dạng:



Từ điển này được chỉ định bởi người dùng bằng cách nhập vào trong textbox và nhấn nút **Load**.

Hướng dẫn: Khi người dùng click nút Load thì tiến hành nạp từ điển trong file vào 2 listbox. Một listbox chứa từ tiếng Anh một listbox chứa từ tiếng Việt. Đọc từng dòng và dùng hàm Split để tách riêng hai từ.

Khi muốn tra cứu thì chỉ việc dùng hàm Contains của Items trong Listbox để xác định từ đó có trong listbox hay không.

Lưu ý: Cần phải có biến listbox tương ứng trong lớp XuLyClient và truyền lstEnglish, lstVietnamese của form chính vào trong thể hiện của lớp XuLyClient để trong lớp XuLyClient có thể truy xuất đến các từ trong hai danh

Dưới đây là chương trình load từ điển vào trong listbox:

```
private void btnLoad_Click(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader(txtDictionaryPath.Text);
    String S, English, Vietnamese;
    String[] Kq;

    lstEnglish.Items.Clear();
    lstVietnamese.Items.Clear();

    while (sr.EndOfStream == false)
    {
        S = sr.ReadLine(); //đọc từng dòng

        Kq = S.Split('#'); //Tách từ Tiếng anh và từ Tiếng việt bị cách nhau bởi dấu '#'
        English = Kq[0].Trim();
        Vietnamese = Kq[1].Trim();

        //Lưu từ tiếng anh và tiếng việt vào trong Listbox tương ứng.
        lstEnglish.Items.Add(English);
        lstVietnamese.Items.Add(Vietnamese);
    }

    sr.Close();
}
```

6.3 Bổ sung thêm tính năng Login cho chương trình. Tức là khi chương trình client muốn tra cứu thì phải thực hiện chức năng kiểm tra mật khẩu.

BÀI THỰC HÀNH SỐ 5

SỬ DỤNG TcpListener ĐỂ XÂY DỰNG ỨNG DỤNG SERVER CÓ SỬ DỤNG GIAO THỨC XÁC THỰC

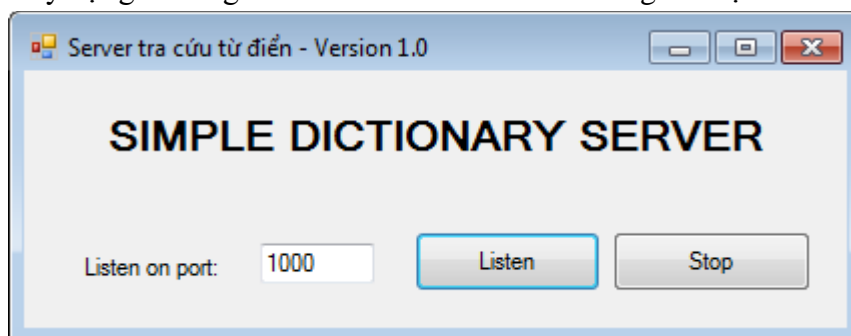
MỤC TIÊU: Kết thúc bài thực hành, sinh viên có thể

Cài đặt được Server tra cứu từ điển có sử dụng một giao thức xác thực đăng nhập đơn giản.

NỘI DUNG:

1. Yêu cầu

Xây dựng chương trình Server tra cứu từ điển có giao diện như sau:



2. Đặc tả chức năng

Khi Client kết nối đến mà chưa thực hiện xác thực (đăng nhập) thì server sẽ không phục vụ các yêu cầu tra cứu.

Để tra cứu được thì client phải tiến hành theo “giao thức” như sau:

B1: Client kết nối đến Server (cổng 1000)

B2: Client gửi user name cho server

B3: Server gửi trả lại chuỗi “200 OK” nếu user tồn tại và “500 User not exist” nếu trái lại.

B4: Client gửi mật khẩu

B5: Server gửi trả lại chuỗi “200 OK” nếu đúng và “501 Password wrong” nếu sai.

B6: Client gửi từ Tiếng Anh

B7: Server gửi trả về nghĩa Tiếng Việt.

3. Hướng dẫn

Bài toán ở đây khác so với các bài thực hành trước ở chỗ có thêm động tác kiểm tra đăng nhập trước khi tiến hành tra cứu từ điển. Trong khi đăng nhập lại có 2 bước theo đúng trình tự là gửi user name trước, sau đó mới gửi mật khẩu.

Vì vậy, đối với server thì khi nhận được gói tin thì nó có thể là:

- User name
- Password
- Từ Tiếng Anh

Khi đó, điều quan trọng nhất đối với server là khi nhận được một gói tin thì phải biết được nó là loại gói tin nào trong 3 dạng trên để từ đó có phương án xử lý phù hợp.

Để giải quyết vấn đề: xác định gói tin client gửi đến đang ở giai đoạn nào, ta có thể sử dụng một biến trạng thái (ví dụ là Bước_Xử_Lý) để ghi nhận giai đoạn xử lý hiện tại. Đầu tiên ta gán nó bằng 0 để hiểu là giai đoạn 1 (nhập user name), tiếp theo gán nó bằng 1 (nhập password), tiếp theo gán nó bằng 2 (nhập từ Tiếng Anh).

Chương trình của chúng ta sẽ căn cứ vào giá trị của biến trạng thái này để so sánh Với user name hoặc password hoặc tiến hành lấy nghĩa Tiếng Việt tương ứng.

4. Chương trình minh họa

Ta viết chương trình này tương tự như bài thực hành số 4, nhưng cần phải thay đổi nội dung của hàm XuLy.

Form1.cs

```
using System;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace TK9LC_TCPServer_V1
{
    public partial class Form1 : Form
    {
        TcpListener Server;
        bool Stop = false;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            lstEnglish.Visible = false;
            lstVietnamese.Visible = false;
        }

        private void btnListen_Click(object sender, EventArgs e)
        {
            //Lắng nghe kết nối tại cổng txtLocalPort, trên tất cả các card mạng.
            Server = new TcpListener(IPAddress.Any, int.Parse(txtLocalPort.Text));
            Server.Start();

            //Đặt lại trạng thái các nút nhấn
            btnStop.Enabled = true;
            btnListen.Enabled = false;
        }
    }
}
```



```

TcpClient Client;
while (!Stop)
{
    if (Server.Pending()) //Nếu có kết nối đến từ client
    {
        //Chấp nhận khi có kết nối đến và lưu kết quả vào biến Client
        Client = Server.AcceptTcpClient();

        //Chuyển Client vừa được tạo ra này cho lớp chuyên xử lý gửi/nhận
        XuLyClient objXuly = new XuLyClient(Client);

        //Chạy phương thức XuLy của ObjXuLy vừa tạo trong 1 tuyến riêng
        Thread Th = new Thread(new ThreadStart(objXuly.XuLy));
        Th.Start();
    }
    Application.DoEvents();
}

private void btnLoad_Click(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader(txtDictionaryPath.Text);
    String S, English, Vietnamese;
    String[] Kq;

    lstEnglish.Items.Clear();
    lstVietnamese.Items.Clear();

    while (sr.EndOfStream == false)
    {
        S = sr.ReadLine(); //đọc từng dòng

        Kq = S.Split('#'); //Tách từ Tiếng anh và từ Tiếng việt
        English = Kq[0].Trim();
        Vietnamese = Kq[1].Trim();

        //Lưu từ tiếng anh và tiếng việt vào trong Listbox tương ứng.
        lstEnglish.Items.Add(English);
        lstVietnamese.Items.Add(Vietnamese);
    }

    sr.Close();
}

private void btnStop_Click(object sender, EventArgs e)
{
    Stop = true;
    Thread.Sleep(100); //Chờ 100ms để cho vòng lặp (while (!Stop)) kết thúc.

    //Stop Server

```

```

Server.Stop();

//Đặt lại trạng thái các nút nhấn cho phù hợp
btnStop.Enabled = false;
btnListen.Enabled = true;
}
}

=====
//Lớp được thiết kế để Server ủy quyền xử lý mỗi kết nối được tạo ra.
class XuLyClient
{
    TcpClient Tcp;
    int Bước_Xử_Lý=0;

    public XuLyClient(TcpClient client)
    {
        Tcp = client; //Lưu tham chiếu đến Client do server đã Accepted
    }

    //Phương thức này sẽ được chạy trong một tuyến riêng biệt
    public void XuLy()
    {
        while (true)
        {
            StreamReader sr = new StreamReader(Tcp.GetStream());
            StreamWriter sw = new StreamWriter(Tcp.GetStream());
            sw.AutoFlush = true;

            String Message, VietNameese;
            Message = sr.ReadLine();

            switch (Bước_Xử_Lý)
            {
                case 0: ///Client gửi xâu đầu tiên (message = user name)
                    if (Message == "tk9lc")
                    {
                        Bước_Xử_Lý = 1; //bước nhập user name đã xong
                        sw.WriteLine("200 OK, mời bạn nhập tiếp mật khẩu.");
                    }
                    else
                        sw.WriteLine("500 Tên người dùng không tồn tại hoặc chưa đăng nhập");
                    break;

                //Trước đó đã nhập đúng user name rồi. Tiếp theo kiểm tra mật khẩu
                case 1:
                    if (Message == "123456")
                    {
                        Bước_Xử_Lý = 2; //Đã xong bước kiểm tra xong mật khẩu.
                        sw.WriteLine("200 OK, Bạn đã đăng nhập thành công.");
                    }
            }
        }
    }
}

```

```

else sw.WriteLine("501 Bạn đã nhập sai mật khẩu!");
break;

//Trước đó đã nhập đúng cả password rồi. Thực hiện dịch bình thường
case 2:
    String English = Message;
    if (English.ToLower() == "hello") VietNameese = "Xin chào!";
    else if (English.ToLower() == "goodbye") VietNameese = "Tạm biệt!";
    else if (English.ToLower() == "computer") VietNameese = "Máy tính";
    else VietNameese = "Không có trong từ điển...";
    sw.WriteLine(VietNameese);
    break;
}
}
}
}
}
}
}

```

5. Nhận xét

- Khi làm việc với các server/Client mà trong đó có thực hiện thứ tự các bước (có giao thức) thì cần phải biết hiện tại ta đang ở bước/giai đoạn nào bằng cách ghi hiện trạng bước xử lý vào một biến trạng thái (Ví dụ chương trình FTP, SMTP, POP3...). Chương trình sẽ căn cứ vào biến trạng thái này để có xử lý phù hợp.
- Ở trên, vì mỗi client khi kết nối đến server có thể ở những trạng thái khác nhau, vì vậy cần phải đặt biến trạng thái thuộc lớp XuLyClient để đảm bảo mỗi kết nối được tạo ra thì sẽ có một biến tương ứng cũng được tạo ra.

6. Bài tập mở rộng

- Viết chương trình Client cho phép người dùng nhập User name và Password vào 2 ô textbox, sau đó chương trình sử dụng tài khoản này và sẽ tự động đăng nhập vào server.
- Viết chương trình SMTP Server theo đúng giao thức chuẩn.