LỜI CẢM ƠN

Để hoàn thành bài luận văn tốt nghiệp này trước hết, em xin cảm ơn các thầy giáo, cô giáo Khoa Công nghệ thông tin Trường Đại học dân lập Hải Phòng những người đã dạy dỗ, trang bị cho em những kiến thức bổ ích trong bốn năm học vừa qua.

Em xin bày tỏ long biết ơn sâu sắc tới thầy Thạc sĩ Đỗ Văn Chiểu, người đã hướng dẫn , chỉ bảo tận tình để em hoàn thành đồ án tốt nghiệp này.

Mục lục

1.1.Lời nói đầu 3
1.2. Mạng nội bộ - LAN (Local Area Netword) 3
1.3. Mô hình Client – Server 4
Chương 2 Tìm hiểu về C# 7
2.1. Tổng quan về C# 7
2.2. Các thành phần cơ bản 7
2.3. Cấu trúc một chương trình C# 13
2.4. Lập trình mạng với C# 13
Sử dụng C# socket 16
2.5. Lập trình với C# Socket helper classes 17
2.6. Lập trình với thread 20
Chương 3. Phân tích và thiết kế chương trình 23
3.1. Cấu trúc chung của chương trình 23
3.2. Phân tích và thiết kế 24
3.2.1 Viết ứng dụng Client (ChatNDraw) 24
3.2.2 Viết ứng dụng Server (PrismServerAdmin) 27
Chương 4. Chương trình thực nghiệm 30
4.1. Giao diện chương trình Server 30
4.2. Giao diện chương trình Client 31
Chương 5. Tổng kết và hướng phát triển của đồ án 33
5.1 Những kết quả đạt được: 33
5.2 Những vấn đề tồn tại 33
5.3 Hướng phát triển của đồ án 33
5.4 Tài liệu Tham khảo 33

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1.Lời nói đầu

Với sự phát triển mạnh mẽ của công nghệ thông tin ngày nay và sự lớn mạnh, rộng khắp của mạng máy tính toàn cầu . Việc ứng dụng tin học vào các lĩnh vực của cuộc sống ngày càng được quan tâm và sử dụng hiệu quả,đem lại lợi ích lớn về mọi mặt trong đời sống. Sự lớn mạnh của mạng máy tính đã xóa bỏ mọi ranh giới về không gian và thời gian để đem con người và tri thức xích lại gần nhau hơn. Thông qua mạng máy tính, con người có thể tiếp xúc với mọi loại tri thức như tri thức văn hóa, xã hội, khoa học kỹ thuật....

Nếu ai đã từng sử dụng Internet chắc sẽ không ít lần nghe hoặc sử dụng dịch vụ chat, đây là dịch vụ khá phổ biến hiện nay, nó cho phép bạn thiết lập các cuộc đối thoại thông qua máy vi tính với người dùng khác trên Internet. Sau khi bạn đã thiết lập được hệ thống này, những gì bạn làm trên máy tính của bạn như gõ chữ, nói chuyện, hình ảnh , truyền dữ liệu thì được hiển thị trên máy kia và ngược lại.

Dịch vụ chat còn đi vào lĩnh vực khác là ứng dụng trong một mạng của công ty có thể là mạng riêng của công ty đó hay mạng Internet. Nó giúp cho các quý giám đốc, những người quản lý không phải tốn nhiều công sức, thời gian khi cần thông báo việc gì đến nhân viên, việc đó có thể là quan trọng, không quan trọng, những vấn đề bí mật,cả đến những vấn đề riêng tư mà mà không sợ các đồng nghiệp khác hoặc cấp dưới biết....

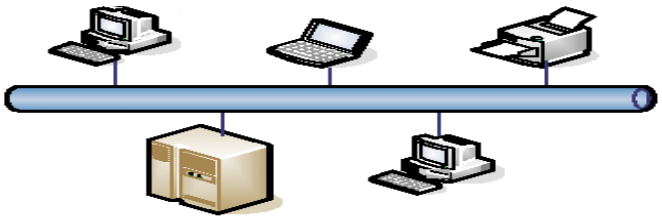
Xuất phát từ lợi ích mà nó đem lại, với mong muốn tạo ra một ứng dụng chat trên mạng LAN, giúp mọi ngươif trao đổi thông tin với nhau lên em chọn đề tài: “ Tìm hiểu về ngôn ngữ C Sharp và viết ứng dụng chát trong mạng LAN ”

1.2. Mạng nội bộ - LAN (Local Area Network)

Mạng nội bộ là một nhóm các máy tính và thiết bị tin học được kết nối với nhau trong một khu vực địa lý nhỏ như một tòa nhà, văn phòng, khuôn viên trường đại học, khu giải trí, ...

Đặc điểm của mạng nội bộ

- *Băng thông lớn có khả năng chạy các ứng dụng trực tuyến như xem phim, hội thảo qua mạng....*
- *Phạm vi bị giới hạn bởi các thiết bị*
- *Chi phí các thiết bị triển khai mạng tương đối rẻ*
- *Dễ quản lý*



Hình 2: Mô hình mạng LAN

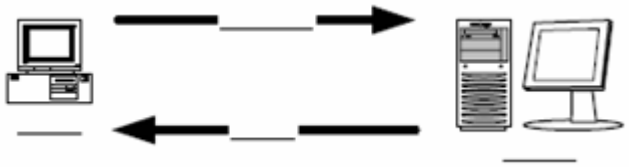
Các mạng LAN trở nên thông dụng vì nó cho phép những người sử dụng (users) dùng chung những tài nguyên quan trọng như máy in , ổ đĩa CD-ROM, các phần mềm ứng dụng và những thông tin cần thiết khác. Bởi vậy đối với những công ty lớn việc mở rộng quy mô hệ thống mạng rất quan trọng nhưng phải đáp ứng yêu cầu trao đổi thông tin một cách nhanh chóng, hiệu quả, tiết kiệm chi phí.

1.3. Mô hình Client – Server

Thuật ngữ server được dùng cho những chương trình thi hành như một dịch vụ trên toàn mạng. Các chương trình server này chấp nhận tất cả các yêu cầu hợp lệ đến từ mọi nơi trên mạng, sau đó nó thi hành dịch vụ và trả kết quả về máy yêu cầu. Một chương trình được coi là client khi nó gửi các yêu cầu tới máy có chương trình server và chờ đợi câu trả lời từ server. Chương trình server và client nói chuyện với nhau bằng các thông điệp (messages) thông qua một cổng truyền thông liên tác IPC (Interprocess Communication). Để một chương trình server và một chương trình client có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn để nói chuyện, chuẩn này được gọi là giao thức. Nếu một chương trình client nào đó muốn yêu cầu lấy thông tin từ server thì nó phải tuân theo giao thức mà server đó đưa ra. Bản thân chúng ta khi cần xây dựng một mô hình client/server

cụ thể thì ta cũng có thể tự tạo ra một giao thức riêng nhưng thường chúng ta chỉ làm được điều này ở tầng ứng dụng của mạng. Với sự phát triển mạng như hiện này thì có rất nhiều giao thức chuẩn trên mạng ra đời nhằm đáp ứng nhu cầu phát triển này. Các giao thức chuẩn (ở tầng mạng và vận chuyển) được sử dụng rộng rãi nhất hiện nay như: giao thức TCP/IP, giao thức SNA của IBM, OSI, ISDN, X.25 hoặc giao thức LAN-to-LAN NetBIOS. Một máy tính chứa chương trình server được coi là một máy chủ hay máy phục vụ (server) và máy chứa chương trình client được coi là máy tớ (client). Mô hình mạng trên đó có các máy chủ và máy tớ giao tiếp với nhau theo 1 hoặc nhiều dịch vụ được gọi là mô hình client/server. Thực tế thì mô hình client/server là sự mở rộng tự nhiên và tiện lợi cho việc truyền thông liên tiến trình trên các máy tính cá nhân. Mô hình này cho phép xây dựng các chương trình client/server một cách dễ dàng và sử dụng chúng để liên tác với nhau để đạt hiệu quả hơn.

Mô hình chuẩn cho các ứng dụng trên mạng là mô hình client-server.Trong mô hình này máy tính đóng vai trò là một client là máy tính có nhu cầu cần phục vụ dịch vụ và máy tính đóng vai trò là một server là máy tính có thể đáp ứng được các yêu cầu về dịch vụ đó từ các client. Khái niệm client-server chỉ mang tính tương đối, điều này có nghĩa là một máy có thể lúc này đóng vai trò là client và lúc khác lại đóng vai trò là server. Nhìn chung, client là một máy tính cá nhân, còn các Server là các máy tính có cấu hình mạnh có chứa các cơ sở dữ liệu và các chương trình ứng dụng để phục vụ một dịch vụ nào đấy từ các yêu cầu của client. Như hình sau:



Mô hình client server

Cách thức hoạt động của mô hình client-server như sau: một tiến trình trên server khởi tạo luôn ở trạng thái chờ yêu cầu từ các tiến trình client, tiến trình tại client được khởi tạo có thể trên cùng hệ thống hoặc trên các hệ thống khác được kết nối thông qua mạng,

tiến trình client thường được khởi tạo bởi các lệnh từ người dùng. Tiến trình client ra yêu cầu và gửi chúng qua mạng tới server để yêu cầu được phục vụ các dịch vụ. Tiến trình trên server thực hiện việc xác định yêu cầu hợp lệ từ lient sau đó phục vụ và trả kết quả tới client và tiếp tục chờ đợi các yêu cầu khác. Một số kiểu dịch vụ mà server có thể cung cấp như: dịch vụ về thời gian (trả yêu cầu thông tin về thời gian tới client), dịch vụ in ấn (phục vụ yêu cầu in tại client), dịch vụ file (gửi, nhận và các thao tác về file cho client), thi hành các lệnh từ client trên server...

CHƯƠNG 2: TÌM HIỂU VỀ C#

2.1. Tổng quan về C#

C# là một ngôn ngữ lập trình hướng đối tượng, cấu trúc và lập luận của C# có đầy đủ của đặc tính của một ngôn ngữ lập trình hướng đối tượng trước đó (C++, Java). C# được thiết kế dung cho nền .Net framework, một công nghệ mới và đầy triển vọng trong việc phát triển các ứng dụng hệ thống và mạng internet…

C# là một trình biên dịch hướng .Net, nghĩa là tất cả các mã của C# luôn luôn chạy trên môi trường .Net Framework. C# là một ngôn ngữ lập tình mới:

- Nó được thiết kế riêng để dùng cho Microsoft's .Net Framework (Một nền khá mạnh cho sự phát triển, triển khai, thực hiện và phân phối các ứng dụng).
- Nó là một ngôn ngữ hoàn toàn hướng đối tượng được thiết kế dựa trên kinh nghiệm của các ngôn ngữ hướng đối tượng khác.

Một điếu quan trọng C# là một ngôn ngữ độc lập. Nó được thiết kế để có thể sinh ra mã đích trong môi trường .Net, nó không phải là một phần của .Net bởi vậy có một vài đặc trưng được hỗ trợ bởi .Net nhưng không hỗ trợ và có những đặc trưng C# hỗ trợ mà .Net không hỗ trợ.

2.2. Các thành phần cơ bản

2.2.1 Biến

+) Một biến dùng để lưu trữ giá trị mang một kiểu dữ liệu nào đó. Cú pháp C# sau đây để khai báo một biến :

[modifier] datatype identifer ;

Với modifier là một trong những từ khoá : public, private, protected, . . . còn datatype là kiểu dữ liệu (int , long , float. . .) và identifier là tên biến.

Để tạo một biến chúng ta phải khai báo kiểu của biến và gán cho biến một tên duy nhất. Biến có thể được khởi tạo giá trị ngay khi được khai báo, hay nó cũng có thể được gán một giá trịmới vào bất cứ lúc nào trong chương trình.

2.2.2 Hằng

Hằng cũng là một biến nhưng giá trị của hằng không thay đổi. Biến là công cụ rất mạnh,tuy nhiên khi làm việc với một giá trị được định nghĩa là không thay đổi, ta phải đảm bảo giá trị của nó không được thay đổi trong suốt chương trình.

2.2.3 Định danh

Định danh là tên mà người lập trình chỉ định cho các kiểu dữ liệu, các phương thức, biến, hằng, hay đối tượng.... Một định danh phải bắt đầu với một ký tự chữ cái hay dấu gạch dưới,các ký tự còn lại phải là ký tự chữ cái, chữ số, dấu gạch dưới.Theo qui ước đặt tên của Microsoft thì đề nghị sử dụng *cú pháp lạc đà* (camel notation) bắtđầu bằng ký tự thường để đặt tên cho các biến là *cú pháp Pascal* (Pascal notation) với ký tự đầu tiên hoa cho cách đặt tên hàm và hầu hết các định danh còn lại

Các định danh không được trùng với các từ khoá mà C# đưa ra, do đó chúng ta không thể tạo các biến có tên như class hay int được. Ngoài ra, C# cũng phân biệt các ký tự thường và ký tự hoa vì vậy C# xem hai biến bienNguyen và bienguyen là hoàn toàn khác nhau.

2.2.4 Kiểu dữ liệu

C# là ngôn ngữ lập trình mạnh về kiểu dữ liệu, một ngôn ngữ mạnh về kiểu dữ liệu là phải khai báo kiểu của mỗi đối tượng khi tạo (kiểu số nguyên, số thực, kiểu chuỗi, kiểu điều khiển...) và trình biên dịch sẽ giúp cho người lập trình không bị lỗi khi chỉ cho phép một loại kiểu dữ liệu có thể được gán cho các kiểu dữ liệu khác. Kiểu dữ liệu của một đối tượng là một tín hiệu để trình biên dịch nhận biết kích thước của một đối tượng

C# chia thành hai tập hợp kiểu dữ liệu chính: Kiểu xây dựng sẵn (built- in) mà ngôn ngữ cung cấp cho người lập trình và kiểu được người dùng định nghĩa(user-defined)

do người lập trình tạo ra. C# phân tập hợp kiểu dữ liệu này thành hai loại: Dữ liệu kiểu trị và kiểu qui chiếu

Nghĩa là trên một chương trình C# dữ liệu được lưu trữ một hoặc hai nơi tuỳ theo đặc thù của kiểu dữ liệu.

Việc phân chia này do sự khác nhau khi lưu kiểu dữ liệu giá trị và kiểu dữ liệu tham chiếu trong bộ nhớ. Đối với một kiểu dữ liệu giá trị thì sẽ được lưu giữ kích thước thật trong bộ nhớ đã cấp phát là stack. Trong khi đó thì địa chỉ của kiểu dữ liệu tham chiếu thì được lưu trong stack nhưng đối tượng thật sự thì lưu trong bộ nhớ heap.

C# cũng hỗ trợ kiểu con trỏ (pointer type) giống như C++ nhưng ít khi dùng đến và chỉ dùng khi làm việc với đoạn mã unmanaged. Đoạn mã unmanaged là đoạn mã được tạo ra ngoài sàn diễn .NET, chẳng hạn những đối tượng COM**.**

Kiểu giá trị được định nghĩa trước (Predefined Value Types)

Kiểu dữ liệu bẩm sinh (The built-in value types) trình bày ban đầu như integer và floating-point numbers, character, và Boolean types.

2.2..5 Câu lệnh

2.2.5.1)Câu lệnh điều kiện

- **Câu lệnh điều kiện if** :

Cú pháp như sau:

if (biểu thức điều kiện)

<Khối lệnh thực hiện khi điều kiện đúng>

[else

<Khối lệnh thực hiện khi điều kiện sai>]

-Câu lệnh switch

Các câu lệnh if nằm lồng rất khó đọc, khó gỡ rối. Khi bạn có một loạt lựa chọn phức tạp thì nên sử dụng câu lệnh switch.

Cú pháp như sau:

switch (biểu thức)
{ casce biểu thức ràng buộc:
câu lệnh
câu lệnh nhảy
[default: câu lệnh mặc định]
}

2.2.5.2)Vòng lặp

C# cung cấp cho chúng ta 4 vòng lặp khác nhau (for, while, do...while, và foreach)cho phép chúng ta thực hiện một đoạn mã lặp lại đến khi đúng điều kiện lặp.

- **Vòng lặp for:**

cú pháp:

for ([phần khởi tạo] ; [biểu thức điều kiện]; [bước lặp])
<Câu lệnh thực hiện>

- **Vòng lặp while (The while Loop)**

Cú pháp như sau :
while (Biểu thức)
<Câu lệnh thực hiện>

- **Vòng lặp do . . . while (The do…while Loop)**

do
<Câu lệnh thực hiện>
while (điều kiện)

-Vòng lặp foreach (The foreach Loop)

Vòng lặp **foreach** cho phép tạo vòng lặp thông qua một tập hợp hay một mảng. Đây là một câu lệnh lặp mới không có trong ngôn ngữ C/C++. Câu lệnh **foreach** có cú pháp chung như sau:

Cú pháp như sau:

foreach (<kiểu tập hợp> <tên truy cập thành phần > in < tên tập hợp>)

<Các câu lệnh thực hiện>

-Câu lệnh goto

-Câu lệnh break

Ta dùng câu lệnh break khi muốn ngưng ngang xương việc thi hành và thoát khỏi vòng lặp.

-Câu lệnh continue

Câu lệnh continue được dùng trong vòng lặp khi bạn muốn khởi động lại một vòng lặp nhưng lại không muốn thi hành phần lệnh còn lại trong vòng lặp, ở một điểm nào đó trong thân vòng lặp.

-Câu lệnh return

Câu lệnh return dùng thoát khỏi một hàm hành sự của một lớp, trả quyền điều khiển về phía triệu gọi hàm (caller). Nếu hàm có một kiểu dữ liệu trả về thì return phải trả về một kiểu dữ liệu này; bằng không thì câu lệnh được dùng không có biểu thức.

2.2.6) Các toán tử

+ Các phép toán số học :+ , - , * , / , % ;

+ Các phép toán logic : & , | , ^, ~ , && ,|| ,! ;

+ Phép cộng chuỗi : + ;

+ Các phép toán tăng giảm: ++ , --;

+ Các phép toán gán : = , += , -= , *= , /= , %=, &= , |= , ^= , <<= , >>= ;

+ Các phép toán quan hệ : ==,!= , < , > , <=, >=;

2.2.7)Lớp

Lớp là một khuôn mẫu thiết yếu mà chúng ta cần tạo ra đối tượng. Mỗi đối tượng chứa dữ liệu và các phương thức chế tác truy cập dữ liệu. Lớp định nghĩa cái mà dữ liệu và hàm của mỗi đối tượng riêng biệt (được gọi là thể hiện) của lớp có thể chứa.

Hàm thành phần (Function Members):

Bao gồm các thuộc tính và các phương thức. Chúng ta sử dụng các từ khoá sau để bổ nghĩa cho một phương thức :

Modifier	**Description**
new	Phương thức ẩn một phương thức kế thừa với cùng kí hiệu
public	Phương thức có thể được truy cập bất kỳ
protected	Phương thức có thể bị truy xuất không từ lớp nó thuộc hoặc từ lớp dẫn xuất;
internal	Phương thức có thể được truy cập không cùng assembly
private	Phương thức có thể được truy cập từ bên trong lớp nó phụ thuộc
Static	Phương thức có thể không được tính trên trên một lớp thể hiển cụ thể
virtual	Phương thức bị ghi đè bởi một lớp dẫn xúât
abstract	Phương thức trừu tượng
override	Phương thức ghi đè một phương thức ảo kế thừa hoặc trừu tượng.
sealed	Phương thức ghi đè một phương thức ảo kế thừa, nhưng không thể bị ghi đè từ lớp kế thừa này
extern	Phương thức được thực thi theo bên ngoài từ một ngôn ngữ khác

Cấu trúc (Structs): Chúng ta sẽ đề cập ngắn gọn là, ngoài các lớp nó cũng có thể để khai báo cho cấu trúc, cú pháp giống như cơ bản bạn biết ngoại trừ chúng ta dùng từ khoá struct thay cho class.

2.2.8)Namespace

Đặc tính *namespace* trong ngôn ngữ C#, nhằm tránh sự xung đột giữa việc sử dụng các thư viện khác nhau từ các nhà cung cấp. Ngoài ra, namespace được xem như là tập hợp các lớp đối tượng, và cung cấp duy nhất các định danh cho các kiểu dữ liệu và được đặt trong một cấu trúc phân cấp. Việc sử dụng namespace trongkhi lập trình là một thói quen tốt, bởi vì công việc này chính là cách lưu các mã nguồn để sửdụng về sau. Ngoài thư viện namespace do MS.NET và các hãng thứ ba cung cấp, ta có thể tạo riêng cho mình các namespace. C# đưa ra từ khóa **using** đề khai báo sử dụng namespace trong chương trình:

using < Tên namespace >

Để tạo một namespace dùng cú pháp sau:

namespace <Tên namespace>

{

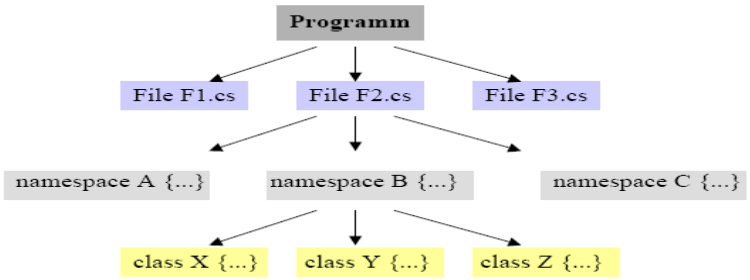
}

< Định nghĩa lớp A>

< Định nghĩa lớp B >

2.3. Cấu trúc một chương trình C#

Một chương trình C# bao gồm các thành phần như sơ đồ dưới đây:



Trong đó :

- Các tệp *.cs là các tệp chứa mã nguồn của một chương trình C#

- Trong mỗi tệp *.cs có các namespace . Nếu không có namespase thì một namespace mặc định được trình biên dịch tự cung cấp. Trong mỗi namespace, có thể có các cấu trúc (structs), các giao diện (Interfaces), các khai báo hằng (enums.

- Trong mỗi namespace, là phần mô tả các lớp đối tượng có trong chương trình

2.4. Lập trình mạng với C#

C# là một ngôn ngữ hỗ trợ lập trình mạng rất mạnh. Trong C#, có rất nhiều lớp đối tượng đã xây dựng sẵn để hỗ trợ lập trình ứng dụng mạng như socket,TCPListener...

Lập trình mạng với socket

Sau đây là những thành phần hỗ trợ lập trình Socket trong C#:

+)Địa chỉ IP trong C#

- IPAddress
- IPEndPoint

2.4.1 IPAddress

Một đối tượng IPAddress được sử dụng để trình bày một địa chỉ IP đơn. Giá trị này sau đó có thể được sử dụng trong một vài phương thức của socket để trình bày địa chỉ IP. Constructor mặc định của lớp này được định nghĩa như sau:

public IPAddress(long address)

Constructor mặc định sẽ nhận một giá trị long và chuyển đổi nó thành một địa chỉ IP. Tuy nhiên trong thực tế phương thức này hầu như không được sử dụng (Vì để đổi một địa chỉ IP thành một số long là công việc khó khăn đối với người lập trình). Thay vào đó một số phương thức khác của IPAddress, được trình bày dưới đây, thường được sử dụng để tạo ra và lưu trữ các địa chỉ IP.

Method	**Description**
Equals	Compares two IP addresses
GetHashCode	Returns a hash value for an IPAddress object
GetType	Returns the type of the IP address instance
HostToNetworkOrder	Converts an IP address from host byte order to network byte order
IsLoopBack	Indicates whether the IP address is considered the loopback address
NetworkToHostOrder	Converts an IP address from network byte order to host byte order
Parse	Converts a string to an IPAddress instance
ToString	Converts an IPAddress to a string representation of the dotted decimal format of the IP address

Phương thức Parse() thường được sử dụng để tạo những thực thể của IPAddress:

```
IPAddress newaddress = IPAddress.Parse("192.168.1.1");
```

Định dạng này cho phép bạn chuyển đổi một chuỗi thể hiện địa chỉ IP theo cách viết thông dụng thành một đối tượng IPAddress.

Lớp IPAddress cũng cung cấp 4 thuộc tính chỉ đọc để trình bày các địa chỉ IP đặc biệt được sử dụng trong chương trình:

Any : Sử dụng để trình bày bất kỳ địa chỉ IP nào có trên hệ thống cục bộ

Broadcast : Sử dụng để trình bày địa chỉ broadcast trong LAN

Loopback : Sử dụng để trình bày địa chỉ loopback trên hệ thống cục bộ

None : Sử dụng để thể hiện không có giao diện mạng trong hệ thống.

2.4.2 IPEndPoint

.NET Framework sử dụng đối tượng IPEndPoint để trình bày một bộ (một đầu socket) IP address/(TCP hoặc UDP)port. Một đối tượng IPEndPoint được sử dụng khi nối kết những sockets tới địa chỉ cục bộ, hoặc khi kết nối sockets tới một địa chỉ ở xa.

Hai constructors được sử dụng để tạo thực thể IPEndPoint:

- IPEndPoint(long *address*, int *port*)
- IPEndPoint(IPAddress *address*, int *port*)

Cả hai constructors sử dụng 2 tham số: một giá trị IP, được thể hiện dưới dạng một giá trị long hoặc một đối tượng IPAddress; và một số hiệu cổng.

Method	**Description**
Create	Creates an EndPoint object from a SocketAddress object
Equals	Compares two IPEndPoint objects
GetHashCode	Returns a hash value for an IPEndPoint object
GetType	Returns the type of the IPEndPoint instance
Serialize	Creates a SocketAddress instance of the IPEndPoint instance
ToString	Creates a string representation of the IPEndPoint instance

Lớp SocketAddress là một lớp đặc biệt trong phạm vi namespace System.Net. Nó thể hiện một dạng khác của một đối tượng IPEndPoint.Lớp này có thể được sử dụng để lưu trữ một thực thể IPEndPoint, để sau đó có thể tạo lại một IPEndPoint bằng phương

thức Create(). Định dạng của SocketAddress là:

- 1 byte trình bày AddressFamily của đối tượng.
- 1 byte thể hiện kích thước của đối tượng.
- 2 bytes là số hiệu cổng của đối tượng.
- Những byte còn lại lưu trữ địa chỉ IP của đối tượng.

Ngoài các phương thức, lớp IPEndPoint còn bao gồm 3 thuộc tính có thể đọc/ghi từ một thực thể:

Address Gets or sets the IP address property
AddressFamily Gets the IP address family
Port Gets or sets the TCP or UDP port number

Các thuộc tính trên có thể được sử dụng với một thực thể IPEndPoint để đạt được các thông tin về các thành phần riêng lẻ của đối tượng IPEndPoint. Những thuộc tính địa chỉ và cổng cũng có thể được sử dụng để thiết lập những giá trị cụ thể trong phạm vi của một đối tượng IPEndPoint đã có.

MaxPort Giá trị cổng lớn nhất có thể chấp nhận được.
MinPort Giá trị cổng nhỏ nhất có thể chấp nhận được.

Sử dụng C# socket

Namespace System.Net.Sockets bao gồm những lớp cung cấp giao diện thực tới các hàm Winsock APIs ở mức thấp.

Socket Construction

Hạt nhân của namespace System.Net.Sockets là lớp Socket. Nó cung cấp cho trình quản lý C# đoạn mã thực thi của Winsock API. Constructor của lớp Socket là:

Socket(AddressFamily *af*, SocketType *st*, ProtocolType *pt*)

Định dạng cơ bản của Socket constructor tương tự như Unix socket. Nó sử dụng 3 tham số để định nghĩa kiểu của socket cần tạo:

- *AddressFamily* để định nghĩa kiểu mạng.

- *SocketType* để định nghĩa kiểu dữ liệu của kết nối.
- *ProtocolType* định nghĩa giao thức mạng

Mỗi tham số trên có thể được sử dụng bằng các hằng được mô tả trong bảng dưới

SocketType	**Protocoltype**	**Description**
Dgram	Udp	Connectionless communication
Stream	Tcp	Connection-oriented communication
Raw	Icmp	Internet Control Message Protocol
Raw	Raw	Plain IP packet communication

Sử dụng những giá trị hằng giúp người lập trình dễ nhớ tất cả các tùy chọn, ví dụ:

```
Socket newsock = Socket(AddressFamily.InterNetwork,
                        SocketType.Stream, ProtocolType.Tcp);
```

Một vài thuộc tính của lớp Socket có thể được sử dụng để lấy ra các thông tin từ một đối tượng Socket đã được tạo. Những thuộc tính đó được mô tả trong bảng dưới:

Property	**Description**
AddressFamily	Gets the address family of the Socket
Available	Gets the amount of data that is ready to be read
Blocking	Gets or sets whether the Socket is in blocking mode
Connected	Gets a value that indicates if the Socket is connected to a remote device
Handle	Gets the operating system handle for the Socket
LocalEndPoint	Gets the local EndPoint object for the Socket
ProtocolType	Gets the protocol type of the Socket
RemoteEndPoint	Gets the remote EndPoint information for the Socket
SocketType	Gets the type of the Socket

2.5. Lập trình với C# Socket helper classes

.NET Framework hỗ trợ giao diện socket nó cũng cung cấp một giao diện đơn giản giúp cho việc lập trình mạng trở nên dễ dàng hơn. Có 3 lớp đối tượng được sử dụng cho việc lập trình với giao diện mới này là:

- TcpClient
- tcpListener
- UdpClient

Mỗi lớp được thiết kế đẻ hỗ trợ một socket đặc biệt phục vụ cho một chức năng của chương trình và giao diện lập trình trở nên đơn giản hơn cho các chức năng đó. Rõ ràng , lớp TcpClient và TcpListener được sử dụng cho việc tạo các chương trình TCP client và server; lớp udp dượ sử dụng để tạo các chương trình UDP

Tcp Client

TCP là một giao thức đáng tin cậy dựa-trên-kết-nối, cho phép hai máy tính giao tiếp thông qua một network. Để tạo một kết nối TCP, một máy tính phải đóng vai trò là server và bắt đầu lắng nghe trên một endpoint cụ thể (endpoint được định nghĩa là một địa chỉ IP, cho biết máy tính và số port). Một máy tính khác phải đóng vai trò là client và gửi một yêu cầu kết nối đến Eendpoint mà máy tính thứ nhất đang lắng nghe trên đó. Một khi kết nối được thiết lập, hai máy tính có thể trao đổi các thông điệp với nhau. Cả hai máy tính chỉ đơn giản đọc/ghi từ một System.Net.Sockets.NetworkStream. Một khi kết nối TCP được thiết lập, hai máy tính có thể gửi bất kỳ kiểu dữ liệu nào bằng cách ghi dữ liệu đó ra NetworkStream

Những phương thức của lớp Tcpclient được sử dụng để tạo ra những chương trình network client theo mô hình hướng kết nối.

+Sử dụng phương thức Connect()

+Sử dụng EndPoint

+Kết nối trực tiếp không cần sử dụng phương thức Connect()

Phương thức getStream() được sử dụng để tạo ra một đối tượng networkStream cho phép bạn gủi và nhận các byte trên socket.Khi bạn có một thực thể NetworkStream cho socket, nó là một sự tách biệt để sử dụng dòng dữ liệu chuẩn để thông qua phương thức Read() và Write() để di chuyển dữ liệu vào và ra trên một socket.

Tcplistener

Lớp TcpClient đơn giản là một client socket dành cho chương trình Client, lớp tcpListener đơn giản là dành cho chương trình server(Một server socket).những constructors của lớp này cũng đơn giản tương tự.Có 3 constructor có khuôn dạng như sau:

- TcpListener(int port) nối kết với một cổng cục bộ
- TcpListener(IPEndPoint ie) kết nối với một EndPoint cục bộ
- TcpListener(IPAddress addr,int port) kết nối tới một IPAddress và cổng cục bộ

Ba khuôn dạng trên cho phép bạn tạo ra các đối tượng *TcpListener*. Khi một đối tượng đã được tạo, bạn có thể bắt đầu lắng nghe cho kết nối mới bằng cách sử dụng phương thức Start().

Sau phương thức Start(), bạn phải sử dụng phương thức AcceptSocket() hoặc AcceptTcpClient() để chấp nhận một yêu cầu kết nối từ Client. Hai phương thức này, như tên của chúng, sẽ cho phép thành lập kết nối và trả về một đối tượng Socket hoặc TcpClient. Chúng ta đã thấy sự thuận tiện khi sử dụng lớp TcpListener, bởi vậy hầu hết chúng ta sẽ sử dụng phương thức AcceptTcpClient() để tạo ra một đối tượng TcpClient mới cho kết nối mới.

Khi một đối tượng TcpClient mới được tạo ra cho kết nối, bạn có thể tận dụng những phương thức chuẩn của TcpClient để bắt đầu giao tiếp với client

UdpClient

Đối với những ứng dụng yêu cầu một socket không hướng nối, lớp UdpClient cung cấp một giao diện đơn giản cho UDP sockets. Chúng ta đều biết rằng UDP là một giao thức không kết nối, do đó nó không giống như ứng dụng client – server, chỉ có những UDP socket đợi để truyền dữ liệu. Chúng ta không cần kết nối UDP socket tới một địa chỉ đặc biệt và chờ nhận dữ liệu.

Constructor của UdpClient cũng có khuông dạng như TcpClient—chúng cho phép ta chỉ định những thông tin cần thiết để tạo một UDP socket hoặc tham chiếu đến một cổng UDP trên một địa chỉ ở xa.

Phương thưc Receive() cho phép bạn nhận dữ liệu trên công UDP. Không có phiên kết nối nào được thành lập, bởi vậy khi một cổng UDP nhận được dữ liệu, bạn không cần biết nó từ đâu đến. Để bù lại cho điều này, phương thức Receive() bao gồm một tham số IPEndPoint chứa thông tin về địa chỉ IP của host ở xa.

Đối tượng UdpClient khi đã được cài đặt sử dụng một cổng UDP mà ứng dụng của bạn muốn để nhận các gói UDP. Chú ý rằng một đối tượng UdpClient trên máy ở xa khi được thiết lập sử dụng giá trị IPAddress.Any. Giá trị này được sử dụng như là một giá trị giả và sẽ được thay thế với thông tin từ phương thức Receive().

Phương thức Send() có 3 khuôn dạng. Nếu đối tượng UdpClient tham chiếu tới một host ở xa, phương thức Send() không cần chỉ định đích của dữ liệu. Tuy nhiên, Nếu đối tượng UdpClient không tham chiếu tới một host ở xa, phương thức Send() phải được chỉ định địa chỉ đích. Việc này có thể thực hiện thông qua một đối tượng IPEndPoint, hoặc một xâu hostname hoặc giá trị IP và một số hiệu cổng

2.6. Lập trình với thread

Khi xây dựng các ứng dụng Server ta luôn muốn nó phải đáp ứng được cho nhiều Client, do đó mỗi ứng dụng Server cần có nhiều SocketClient và chúng ta phải thực hiện chúng đồng thời. Nhưng phương thức Receive() trên mỗi SocketClinet sẽ rơi vào trạng thái chờ đợi khi không có dữ liệu kết nối mà nó quản lý và vì vậy khi một kết nối đang được xử lý không có dữ liệu các kết nối khác sẽ bị khóa. Để giải quyết vấn đề này C# cung cấp một công cụ lập trình Thread.

Lớp .Net processThread trong namespace System.Diagnostics cho phép bạn truy nhập các thread trong phạm vi một tiến trình từ chương trình C# của bạn. Lớp processThread bao gồm nhiều thuộc tính và phương thức được sử dụng để đạt được thông tin về các thread trong một tiến trình thực thi.

Các tiến trình tạo ra một hoặc nhiều *threads*. Một thread xác định một luồng đơn thực thi trong phạm vi một chương trình. Khi một chương trình thực thi trên CPU, nó thực hiện các câu lệnh chương trình trong một luồng đơn tới khi luồng hoàn thành. Cho phép hai hoặc nhiều tác vụ cùng thực hiện. Mỗi một tác vụ được thực thi trên một luồng riêng.

Chương trình multithreaded tạo ra các luồng của nó và có thể thực thi các luồng đó từ luồng chính của chương trình. Tất cả các luồng được tạo ra từ luồng chính bằng cách chia sẻ không gian nhớ của luồng chính. Thông thường luồng thứ hai được sử dụng để thực hiện các công việc tính toán của chương trình, cho phép luồng chính phản ứng với các sự kiện. Không có luồng thứ 2, người sử dụng không thể lựa chọn thực đơn hoặc bất kỳ nút lệnh nào khi một chương trình đang tính toán các hàm toán học.

Threads có một vài trạng thái thực hiện, những trạng thái đó được định nghĩa sẵn bằng các hằng trong.Net. Các hằng đó được định nghĩa trong namespace System.Threading. Bảng sau liệt kê những trạng thái có thể có của một thread trong hệ điều hành Windows.

ThreadState	**Description**
Initialized	The thread has been initialized but not started.
Ready	The thread is waiting for a processor.
Running	The thread is currently using the processor.
Standby	The thread is about to use the processor.
Terminated	The thread is finished and is ready to exit.
Transition	The thread is waiting for a resource other than the processor.
Unknown	The system is unable to determine the thread state.
Wait	The thread is not ready to use the processor.

Một thread đơn lẻ có thể phải thay đổi trạng thái tại một vài thời điểm trong quá trình chạy của nó, chuyển đổi giữa trạng thái Running và Standby, Transition, hoặc Wait. Tự bản thân hệ điều hành có thể đặt một thread đang chạy chuyển sang một trạng thái khác để thực thi một thread khác có độ ưu tiên cao hơn.

Lớp .NET Framework's ProcessThread

Lớp .NET ProcessThread trong namespace System.Diagnostics cho phép bạn truy nhập các thread trong phạm vi một tiến trình từ chương trình C# của bạn. Lớp ProcessThread bao gồm nhiều thuộc tính và phương thức được sử dụng để đạt được thông tin về các thread trong một tiến trình. Bảng sau chỉ ra các thuộc tính của ProcessThread sẵn có để đạt được thông tin về luồng thực thi.

Property	**Description**
BasePriority	Gets the base priority of the thread
Container	Gets the IContainer that contains the thread object
CurrentPriority	Gets the current priority of the thread
Id	Gets the unique thread identifier of the thread
IdealProcessor	Sets the preferred processor for the thread to run on
PriorityBoostEnabled	Gets or sets a value indicating whether the operating system should temporarily boost the priority of the thread when its main window has the focus
PriorityLevel	Gets or sets the priority level of the thread
PrivilegedProcessorTime	Gets the amount of time the thread has spent running code in the operating system core
ProcessorAffinity	Sets the processors that the thread can run on
Site	Gets or sets the ISite of the thread object
StartAddress	Gets the memory address of the function used to start the thread
ThreadState	Gets the current state of the thread
TotalProcessorTime	Gets the total amount of time the thread has spent using the processor
UserProcessorTime	Gets the amount of time the thread has spent running code in the application
WaitReason	Gets the reason the thread is in the Wait state

CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ CHƯƠNG TRÌNH

3.1. Cấu trúc chung của chương trình

Chương trình Chat được chia làm 3 project sau:

- SCG.Prism : Chứa các thành phần và các lớp gói gọn của chương trình như kết nối, giao diện …

Project này bao gồm các thành phần và các lớp quan trọng sau:

1. Thành phần PrismConnection – Cho phép một ứng dụng client kết nối và giao tiếp với một máy chủ. Quy định rõ các đặc tính như địa chỉ máy chủ và số cổng của máy chủ, những phương thức để bắt đầu truyền thông, và những sự kiện để đáp ứng những hoạt động khác nhau như : vào chat, đến một người dùng mới vào một phòng chat, hoặc một thông điệp từ quản trị viên
2. Thành phần PrismServer- Gói gọn các hoạt động của PrismServer, cho phép nhiều client kết nối và giao tiếp thông qua những socket. Đây là thành phần cơ bản của bất cứ ứng dụng máy chủ PrismServer nào, và cung cấp những thuộc tính của giao diện máy chủ người dùng vẫn còn được cập nhật.
3. Lớp PrismUser – Đại diện cho một cá nhân riêng là người đã đăng nhập vào PrismServer. Lớp này được sử dụng để đại diện cho người dùng trong client và ứng dụng server.
4. Lớp PrismRoom – Đại diện cho một "phòng chat" để có thể chứa một hoặc nhiều PrismUser người dùng có thể giao tiếp với nhau.

- PrismServerAdmin : Đóng vai trò là máy Server, có nhiệm vụ lắng các kết nối từ Client thông qua cổng. Cung cấp giao diện người dùng để máy chủ có thể khai thác nhìn thấy ai đã kết nối và hoạt động của màn hình, lịch sử thực thi.
- ChatNDraw: Có nhiệm vụ kết nối đến Server thông qua địa chỉ IP của Server, tạo và vào phòng chat, chat với những người dùng khác trong phòng chat.

3.2. Phân tích và thiết kế

3.2.1 Viết ứng dụng Client (ChatNDraw)

Một ứng dụng Client có thể kết nối và giao tiếp qua một PrismServer, sử dụng thành phần PrismConnection. Trong một ứng dụng WinForms, bạn có thể chỉ cần thả các thành phần này vào các ứng dụng của biểu mẫu và thiết lập các đặc tính trong việc thiết kế thời gian. PrismConnection công bố ba đặc tính đó phải được đặt trước khi kết nối:

1. Host –Ghi rõ tên miền hoặc địa chỉ IP của máy chủ nơi mà PrismServer đang chạy
2. Port –Ghi rõ số cổng mà PrismServer đã cấu hình để sử dụng trong máy chủ
3. SubjectName –Khái niệm "SubjectName" cung cấp khả năng cho nhiều ứng dụng client để sử dụng cho một PrismServer đơn, và chỉ nhận được các thông điệp đã được bắt nguồn từ cùng một client. PrismServer sẽ hướng các thông điệp chỉ để kết nối với những Client mà dung chung cùng một chủ đề. Bạn nên cung cấp một chuỗi giá trị tương ứng với tên ứng dụng của bạn

a. Kết nối

Các đặc tính được thiết lập ở trên, thiết lập đặc tính các hoạt động cho đúng để cố gắng thiết lập một kết nối. Hành động này sẽ chặn các ứng dụng cho đến khi một kết nối được tạo ra, hoặc một ngoại lệ được bỏ lại .

b.Đăng nhập

Khi mà một kết nối được thiết lập,bước tiếp theo là "đăng nhập " vào PrismServer bằng cách sử dụng một tài khoản/mật khẩu. Thành phần PrismConnection cung cấp hai phương pháp có thể được sử dụng để đăng nhập vào máy chủ: Login và LoginNew. Cho phép những người sử dụng để đăng nhập vào máy chủ sử dụng tên người dùng và mật khẩu có sẵn, hoặc là đăng ký mới. Giao diện người dùng cần của client sẽ cung cấp đường dẫn để cho người sử dụng có thể bắt đầu mỗi loại hình đăng nhập.

Gọi Login để đăng nhập vào máy chủ sử dụng tài khoản và mật khẩu đã đăng ký trước đó. PrismConnection sẽ đáp lại thông qua sự kiện LoginOK, hoặc sự kiện

Loginerror. Có thể có các lỗi là mật khẩu không hợp lệ, và không tìm thấy tài khoản người dùng trong đăng ký của máy chủ.

Gọi LoginNew để đăng nhập vào máy chủ với tài khoản mới. Phương pháp này thực thi như một tham số đối tượng PrismUser là một ví dụ, trong đó chứa đựng các thông tin người dùng cho người sử dụng mới. Các thành phần trong thư viện có chứa thành phần hộp thoại PrismUserInfoDialog có thể được sử dụng để tạo ra PrismUser một cách nhanh chóng. PrismConnection sẽ đáp lại, hoặc thông qua các sự kiện LoginOK, hoặc sự kiện Loginerror. Có thể tồn tại các lỗi bao gồm tài khoản đã tồn tại trên danh nghĩa hoặc một tài khoản không hợp lệ.

Ứng dụng trong máy client, bạn luôn có thể tìm ra được ví dụ như PrismUser cục bộ để được kết nối bằng cách tham khảo đặc tính ThisUser.

```
//Đoạn mã ví dụ Client kết nối PrismServer
        connection.Host = _frmLogin.txtHost.Text;
        connection.Port = (int)_frmLogin.numPort.Value;
        try
        {
          connection.Active = true;
          connection.Login(_frmLogin.txtUserName.Text, _frmLogin.txtPassword.Text);
        }
        catch (Exception error)
        {
          MessageBox.Show(error.Message, " Lỗi kết nối PrismServer ");
        }
```

c. PrismRooms

Mỗi một máy client kết nối tới một PrismServer chiếm lĩnh một PrismRoom riêng và có thể chat và tương tác với các máy client khác trong cùng một phòng. Ứng dụng trên máy client của bạn có thể nhìn thấy tất cả các ứng dụng của các PrismRoom đã được tạo ra mà chia sẻ các chủ đề

Thành phần PrismConnection thông báo cho máy client biết PrismRooms được tạo ra gần đây thông qua sự kiện RoomAdded. Bạn sẽ nhận được một loạt các sự kiện này ngay lập tức sau khi đăng nhập thành công vào mỗi PrismRoom hiện tại đang tồn tại trên máy

chủ. Máy client sẽ đại diện cho các phòng trong các giao diện người dùng, bằng cách sử dụng một ListBox, ListView, hoặc điều khiển tương tự. Khi một PrismRoom được lấy ra, PrismConnection thông báo cho máy client thông qua một sự kiện RoomRemoved. PrismConnection cũng thông báo cho máy client khi người sử dụng vào và rời PrismRoom. Thông qua các sự kiện UserAddedToRoom, UserLeftRoom đã hoàn thành xong.

Sau khi đăng nhập thành công vào máy client được đặt vào phòng mặc định hoặc phòng ngoài. PrismConnection thông báo cho máy client là nó đã nhập vào một phòng mới qua sự kiện JoinedRoom. Lưu ý rằng JoinedRoom được kích hoạt, thêm vào một sự kiện accompanyingUserAddedToRoom; UserAddedToRoom có chứa các đối tượng PrismUser là đại diện cho các máy client có nghĩa là thực sự kết nối trong ứng dụng cục bộ .

Để tham gia PrismRoom khác nhau, gọi phương thức EnterRoom. Để tạo tạo một PrismRoom mới, gọi phương thức CreateRoom. Sau khi gọi một trong những phương thức, bạn sẽ nhận được một vài sự kiện, UserLeftRoom (Bạn đã rời khỏi phòng), UserAddedToRoom (cho rằng bạn đã nhập phòng mới), và cuối cùng JoinedRoom. Ngoài ra, nếu bạn là người dùng cuối cùng rời một phòng, PrismServer sẽ hủy phòng (trừ khi nó được mặc định phòng ngoài) và PrismConnection sẽ kích hoạt sự kiện RoomRemoved. Nếu bạn tạo một phòng mới, bạn sẽ nhận được những sự kiện RoomAdded trước khi UserAddedToRoom và JoinedRoom.

d. PrismRooms dành cho những trò chơi nhiều người chơi

Phương thức CreateRoom có 2 tham số, lệnh tên phòng và số lượng tối đa mà người tham gia phòng có thể chứa (0 để chỉ định cho biết không có tối đa số). Nếu bạn vượt qua một số lớn hơn số không, PrismConnectionwill kích hoạt một sự kiện StartSignal khi các chỉ số lượng người tham gia đã nhập các phòng, và máy chủ sẽ khóa phòng như vậy không cho máy client khác có thể vào phòng. Tính năng này được thiết kế để cho phép nhiều người chơi trò chơi để bắt đầu sau khi xác định số lượng người chơi đã nhập trò chơi phòng.

e. Những thông điệp Chat và Dữ liệu

Khi bạn ở trong một PrismRoom ,bạn có thể trao đổi chat và những thông điệp dữ liệu với những máy client khác trong phòng. Để gửi một chuỗi chat văn bản, gọi phương thức SendChat. Khi một chuỗi của chat văn bản được nhận từ máy client khác trong phòng (lưu ý điều này loại trừ máy cục bộ), PrismConnection kích hoạt sự kiện ChatMessageReceived

"Data Messages" cũng là những chuỗi đã được gủi tới những máy client trong phòng, và được quản lý bộ nhớ tương tự như cách để chatstrings. Ý tưởng đằng sau thông điệp dữ liệu là để cung cấp điều kiện thuận lợi cho những ứng dụng trên máy client thông qua ứng dụng cụ thể, không thể chat, dữ liệu cho các máy client trong phòng. Gọi SendData để gửi thông điệp dữ liệu, và đáp ứng với các thông điệp dữ liệu bởi xử lý các sự kiện DataMessageRecieved.

Tóm lại: ChatNDraw bao gồm mã nguồn đầy đủ cho ứng dụng đơn giản trên máy client . Điều này cho phép người dùng trên máy client kết nối với một PrismServer, tạo và tham gia phòng chat, chat với những người dùng khác, và chia sẻ vẽ trên một bảng đen. Cho phép người dùng sửa đổi thông tin cá nhân, xem trạng thái của Server..

3.2.2 Viết ứng dụng Server (PrismServerAdmin)

Các thành phần PrismServer là cơ sở của một PrismServer ứng dụng máy chủ. Gói gọn đa chuỗi server socket, quản lý các Subject Names, PrismRooms, và PrismUsers và định tuyến xử lý các thông điệp đã được kết nối với những máy client. Thành phần PrismServer cung cấp một số đặc tính để điều khiển các mặt hoạt động của máy chủ:

- Port – Chỉ rõ số cổng mà máy chủ sẽ lắng nghe cho các kết nối trên.
- LobbyName –Tên mặc đinh PrismRoom mà những máy client mới thêm vào.
- ProhibitSameIP – Nếu đúng, cấm nhiều kết nối từ những địa chỉ IP giống nhau.
- ProhibitSameUserName – Nếu đúng, cấm nhiều người dùng đăng nhập cùng một tên

- PingInterval – Kiểm soát những Client ping vào Server, những client ping không được đáp ứng như đã ra ngoài và không kết nối.
- Implementation – Cần thiết để thiết lập một số trường hợp của thành phần mà xuất phát từ PrismServerImplementation

Thành phần PrismServer cũng cung cấp một vài phương thức để mà cho phép điều hành viên tương tác giữa server và kết nối với những client, như một chuỗi các sự kiện. Những ứng dụng nên đáp lại những sự kiện được cập nhật từ giao diện người dùng để phản hồi lại thông tin đã thay đổi (Những client mới kết nối, thêm hoặc xóa phòng…). Quan sát ứng dụng PrismServerAdmin bao gồm ứng dụng server mẫu để thực thi những hành động.

Quản lý User trong Server

Thành phần PrismServer trình bày một đặc tính Implementation mà phải gán cho một thành phần mà xuất phát từ PrismServerImplementation. PrismServerImplementation cung cấp một giao diện cho việc quản lý người dùng. Bạn có thể bắt nguồn từ thành phần PrismServerImplementation mới để cho phép server sử dụng tệp cục bộ, một cơ sở dữ liệu, hoặc một vài kỹ thuật lưu trữ khác để quản lý thông tin người dùng. Kỹ thuật lưu trữ được định nghĩa bắt nguồn từ lớp để sau đó tham khảo như "Đăng ký thông tin người dùng" điều này không phù hợp với đăng ký của Window. Bao gồm những gói cụ thể mà bạn có thể sử dụng : PrismServerFileImplementation. Đây là thành phần lưu trữ thông tin người dùng bằng file nhị phân trong cục bộ file hệ thống.

Thành phần PrismServerImplementation chứa những phương thức mà quan trọng đưa ra việc bổ sung quản lý người dùng.

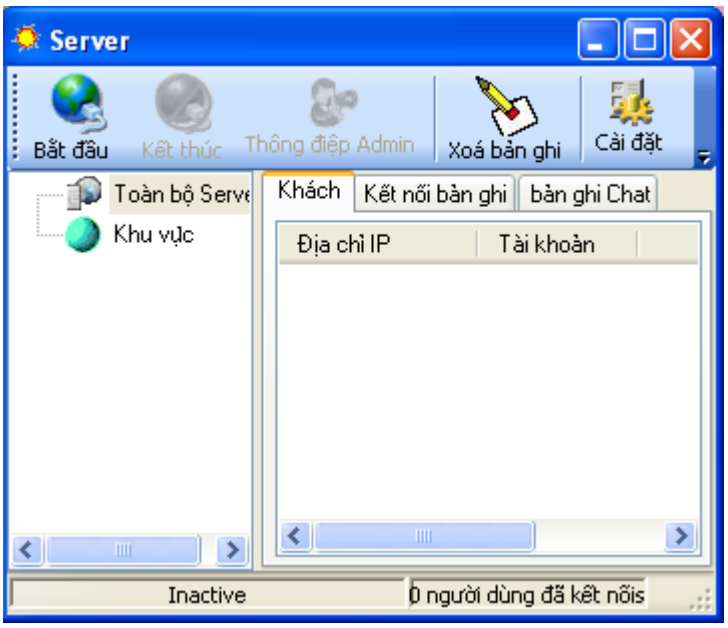
- bool UserExists(string userName) - Trả lại tên tài khoản đã tồn tại trong khi người dùng đăng ký.
- bool IsPasswordValid(string userName, string password) – Hiệu chỉnh lại mật khẩu cho người dùng
- bool CheckUserName(string userName, ref string msg) – Kiểm tra tên tài khoản
- bool CheckPassword(string password, ref string msg) – Kiểm tra mật khẩu

- void StoreUserInfo(PrismUser user) – Lưu trữ thông tin người dùng khi đăng ký.
- bool CheckRoom(string roomName, int maxUsers, ref string msg) – Kiểm tra phòng.
- void SaveSettings() – Ghi lại những thiết lập của Server.
- void LoadSettings() – Nạp vào những thiết lập của Server..
- void Initialize() – Thực hiện khởi tạo trước đây..
- void ProcessCustomCommand(string commandName, string commandParams) – Cung cấp một kỹ thuật để xử lý những lệnh tùy chỉnh của Server. Một ứng dụng Client có thể gọi phương thức CustomCommand của PrismConnection để gửi những lệnh tùy chỉnh tới Server. Một lệnh tùy chỉnh gồm có tên lệnh (string) và những tham số (string). Thành phần PrismServer cũng có thể chuyển cho client những lệnh tùy chỉnh bằng cách sử dunhj phương thức CustomCommand.Cả hai thành phần CustomCommand và PrismConnection đều cung cấp những sự kiện CustomCommandRecieved để xác nhận những lệnh tùy chỉnh. Đây là cấu trúc rất hay đó là tính mềm dẻo và tình tùy biến đối với Client và Server

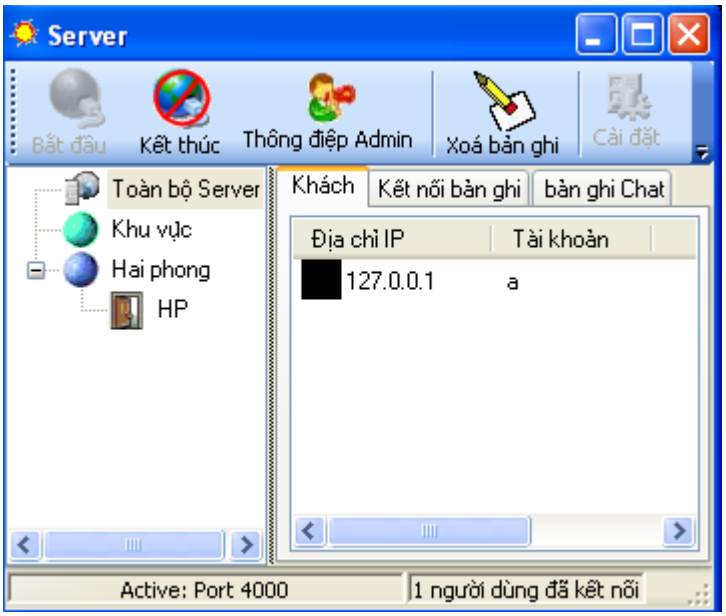
CHƯƠNG 4. CHƯƠNG TRÌNH THỰC NGHIỆM

4.1. Giao diện chương trình Server

Giao diện chương trình Server khi chưa có một Client nào kết nối đến Server. Muốn Server lắng nghe ta kích hoạt bắt đầu, và kết thúc thì sử dụng nút Kết thúc.

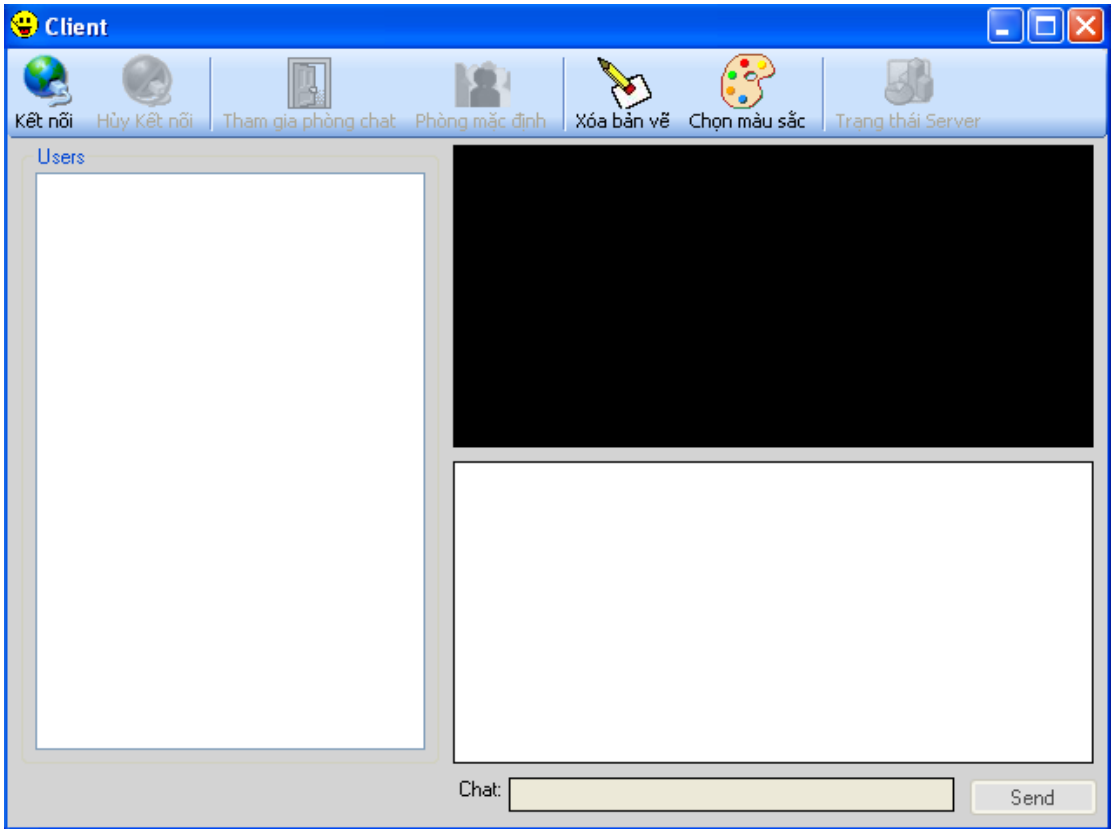


Đây là giao diện server khi bắt đầu lắng nghe



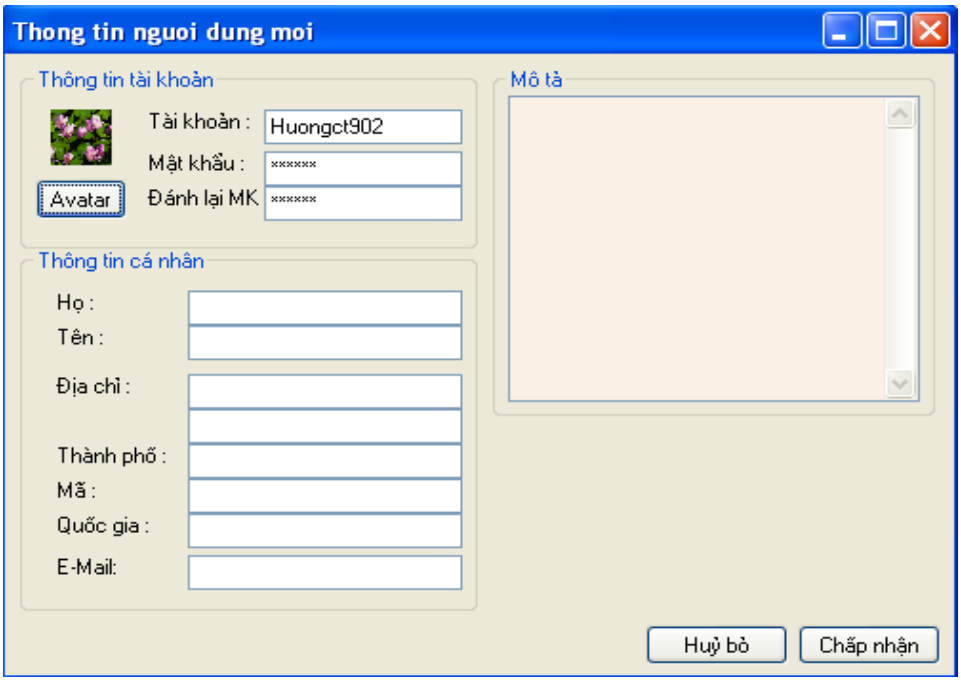
4.2. Giao diện chương trình Client

Giao diện Client khi chưa kết nối

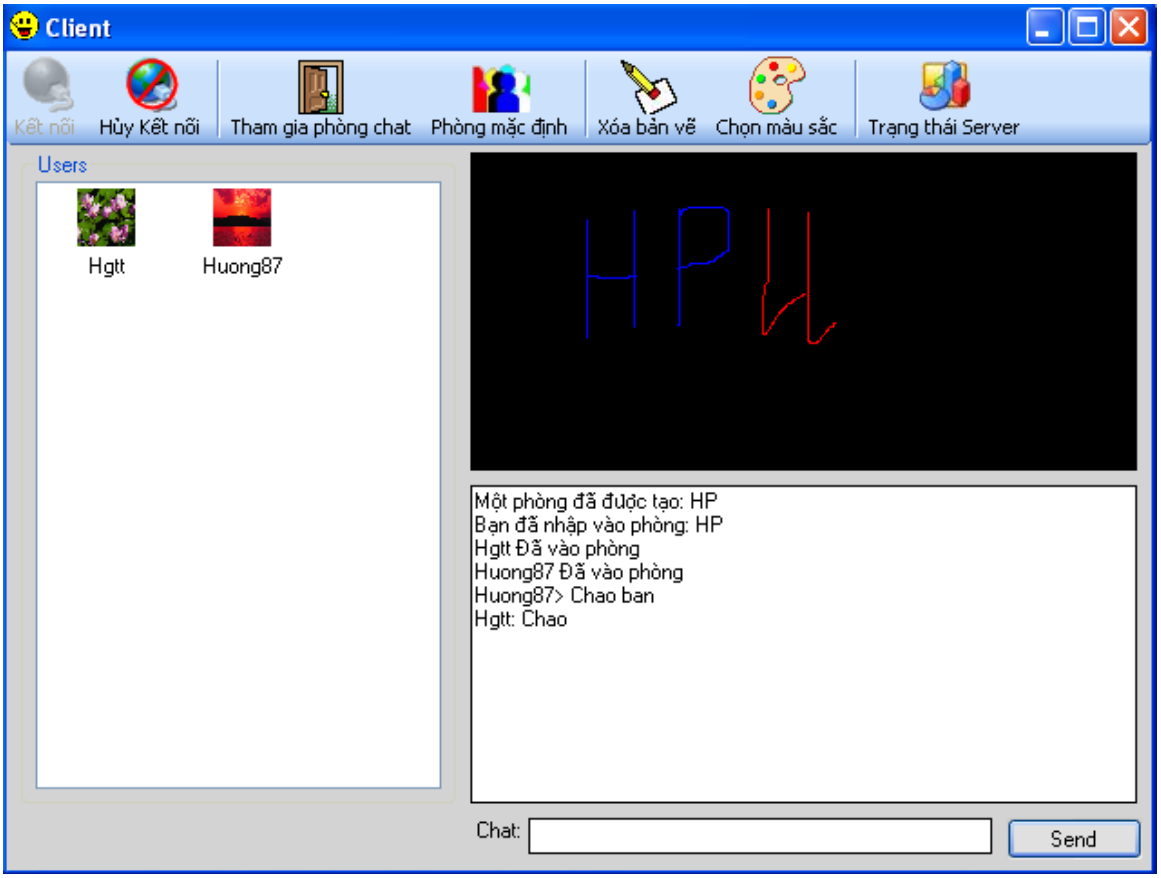


Khi Client muốn kết nối người dùng phải nhấn vào nút “Kết nối”. Sau đó điền các thông tin vào bảng sau :

Nếu chưa đăng ký Tài khoản thì nhấn “ Tạo tài khoản ” để thực hiện đăng ký :



Giao diện người dùng sau khi kết nối thành công:



CHƯƠNG 5. TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN ĐỒ ÁN

5.1 Những kết quả đạt được

Luận văn đã đạt được những kết quả sau:

- Xây dựng thành công ứng dụng Chát trên mô hình Client – Server cho phép nhiều người có thể chat với nhau trên mạng Lan.
- Chương trình chạy tương đối ổn định.
- Thực hiện được chức năng quản lý User .
- Tìm hiểu ngôn ngữ lập trình C#.

5.2 Những vấn đề tồn tại

- Chưa thực hiện Chat được bằng tiếng việt.
- Các tiện ích của hệ thống chưa nhiều như Voice và CAM.

5.3 Hướng phát triển của đồ án

Để hệ thống có thể thực sự giúp nhiều tiện ích cho người dùng, cần phải cải tiến , bổ sung và khắc phục những yếu kém, những vấn đề còn tồn tại của hệ thống.

Thêm và tích hợp một số chức năng tiện ích khác.

5.4 Tài liệu Tham khảo

[1] Richard Blum, C# Network Programming, Sybex © 2003, ISBN:0782141765

[2] Kỹ thuật lập trình C#, Biên dịch từ cuốn Professional C#, 2nd Edition, Xuất bản bởi Wrox Press Ltd .

[3] Andy Harris, Microsoft C# Programming for the Absolute Beginner, Premier Press © 2002, ISBN: 1931841160

[4] Kỹ thuật lập trình ứng dụng C#.net toàn tập ,biên soạn Phạm Hữu Khang ,NXB Lao động – Xã hội , xuất bản năm 2005.

[5] Mạng máy tính , tác giả Lê Đình Danh.