**LẬP TRÌNH MẠNG**

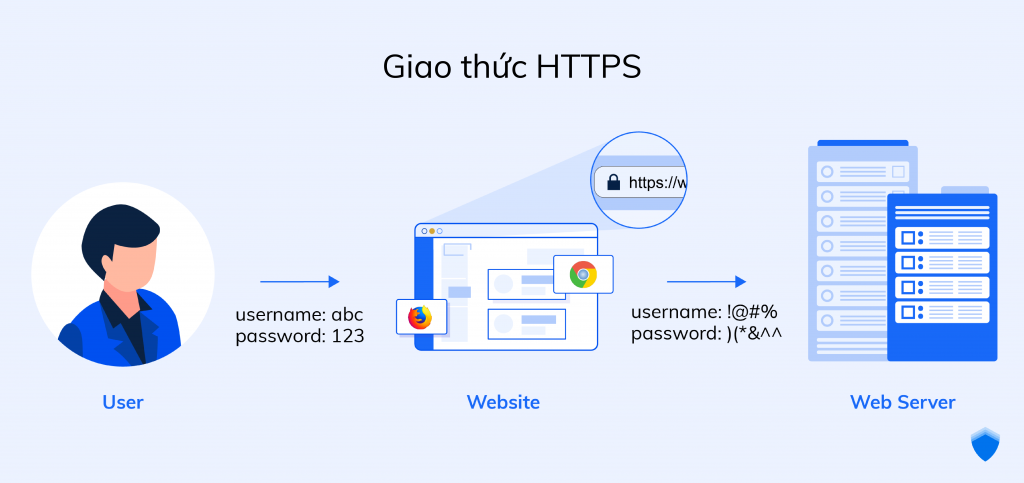
Nhóm OTTI:

1. Phan Văn Thông
2. Zơ Râm Guân
3. Nguyễn Hoàng Lân
4. Thượng Thị Thu Hiệp.

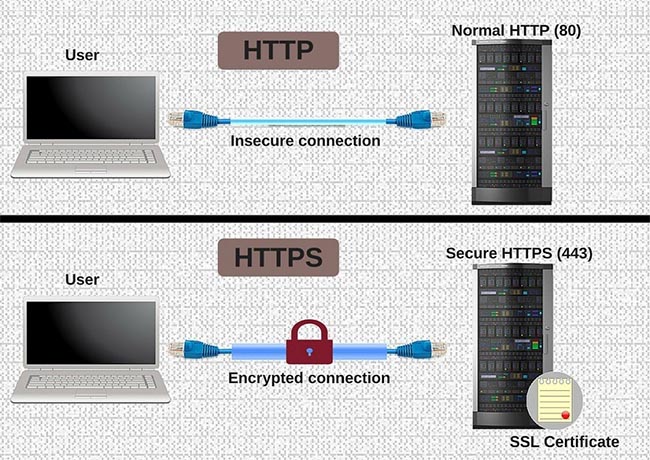
**BÁO CÁO**

1. **Tìm đọc và viết báo cáo ngắn về giao thức HTTP,HTTPS. So sánh hai giao thức này.**

* Giao thức HTTP, HTTPS:
* **HTTP (HyperText Transfer Protocol)** là giao thức tải siêu văn bản sử dụng trog www dùng để truyền tải dữ liệu giữa Web server đến các trình duyệt Web và ngược lại. Giao thức này sử dụng cổng 80 (port 80) là chủ yếu.HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP(các giao thức nền tảng cho Internet). Bộ giao thức TCP/IP là một bộ các giao thức truyền thông cái đặt chồng giao thức mà Internet và hầu hết các mạng máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt theo tên hai giao thức chính là TCP(Transmission Contro Protocol- Giao thức điều khiển truyền vận) và IP(Internet Protocol- Giao thức Internet).Hay bạn có thể hiểu khi bạn gõ vào 1 địa chỉ vào dình duyệt Web, lúc này trình duyệt Web sẽ gửi 1 yêu cầu qua giao thức Http đến Web server, Web server và sẽ nhận yêu cầu này và trả lại kết quả cho trình duyệt Web.
* HTTP hoạt động trên mô hình Client (máy khách) –Server (máy chủ). Các máy khách sẽ gửi yêu cầu đến máy chủ và chờ sự hồi đáp của máy chủ. Để có thể trao đổi thông tin được với nhau, các máy chủ và máy khách phải thực hiện trên một giao thức thống nhất, đó chính là HTTP.Khi bạn truy cập một trang web qua giao thức HTTP, trình duyệt sẽ thực hiện các phiên kết nối đến server của trang web đó thông qua địa chỉ IP do hệ thống phân giải tên miền DNS cung cấp. Máy chủ sau khi nhận lệnh, sẽ trả về lệnh tương ứng giúp hiển thị website, bao gồm các nội dung như: văn bản, ảnh, video, âm thanh,…Trong quá trình kết nối và trao đổi thông tin, trình duyệt của bạn sẽ mặc nhiên thừa nhận địa chỉ IP đó đến từ server của chính website mà bạn muốn truy cập mà không hề có biện pháp xác thực nào. Các thông tin được gửi đi qua giao thức HTTP (bao gồm địa chỉ IP, các thông tin mà bạn nhập vào website…) cũng không hề được mã hóa và bảo mật. Đây chính là kẽ hở mà nhiều hacker đã lợi dụng để đánh cắp thông tin người dùng, thường được gọi là tấn công sniffing.
* **HTTPS (HyperText Transfer Protocol Secure**) là giao thức HTTP có sử dụng thêm SSL(Secure Sockets Layer) để mã hóa dữ liệu trong lúc truyền tải dữ liệu nhầm gia tăng thêm tính an toàn cho việc truyền dữ liệu giữa Web server và trình duyệt Web. Giao thức HTTPS thì sử dụng cổng 443 để truyền dữ liệu.
* **HTTPS** là giao thức truyền tải siêu văn bản an toàn. Thực chất, đây chính là giao thức HTTP nhưng tích hợp thêm Chứng chỉ bảo mật SSL nhằm mã hóa các thông điệp giao tiếp để tăng tính bảo mật. Có thể hiểu, HTTPS là phiên bản HTTP an toàn, bảo mật hơn.



* HTTPS hoạt động tương tự như HTTP, tuy nhiên được bổ sung thêm chứng chỉ **SSL (Secure Sockets Layer – tầng ổ bảo mật)** hoặc **TLS (Transport Layer Security – bảo mật tầng truyền tải)**. Hiện tại, đây là các tiêu chuẩn bảo mật hàng đầu cho hàng triệu website trên toàn thế giới.
* Hiện nay thì việc sử dụng giao thức HTTPS chủ yếu được dùng cho các trang web có giao dịch trực tuyến sử dụng thẻ thanh toán đơn hàng. Nhằm đảm bảo an toàn cho giao dịch, tránh những rủi ro bị lấy mất thông tin thẻ trong quá trình thanh toán.
* Ngoài ra web sử dụng giao thức HTTPS là một tín hiệu để Google xếp hạng từ khóa. Việc này khiến rất nhiều trang Web chuyển qua dùng giao thức HTTPS để SEO được tốt hơn.
* Nhưng hiện nay, Chrome sắp có cảnh báo vưới những website còn sử dụng HTTP, và ưu tiên cho các trang web sử dụng HTTPS.
* Cả SSL và TLS đều sử dụng hệ thống PKI (Public Key Infrastructure -hạ tầng khóa công khai) không đối xứng. Hệ thống này sử dụng hai “khóa” để mã hóa thông tin liên lạc, “khóa công khai” (public key) và “khóa riêng” (private key). Bất cứ thứ gì được mã hóa bằng khóa công khai chỉ có thể được giải mã bởi khóa riêng và ngược lại. Các tiêu chuẩn này đảm bảo các nội dung sẽ được mã hóa trước khi truyền đi, và giải mã khi nhận. Điều này khiến hacker dù có chen ngang lấy được thông tin cũng không thể “hiểu” được thông tin đó.
* Nói dễ hiểu hơn khi bạn nhập một địa chỉ web và ấn Enter, một lệnh HTTP sẽ được gửi lên máy chủ để yêu cầu tìm website bạn đã nhập. Sau khi máy chủ nhận được yêu cầu, nó sẽ trả lại tìm đến website được yêu cầu đó, và trả lại kết quả cho bạn bằng việc hiển thị website đó lên trình duyệt web của bạn. Quá trình này diễn ra nhanh hay chậm tùy thuộc vào tốc độ Internet của bạn.
* **So sánh giao thức** HTTP, HTTPS:



**Bảng so sánh sự khác biệt giữa HTTP và HTTPS**

|  |  |  |
| --- | --- | --- |
|  | **HTTP** | **HTTPS** |
|  | HTTP là viết tắt của " HyperText Transfer Protocol ". | HTTPS là viết tắt của "Giao thức truyền HyperText qua SSL ( Secure Socket Layer) " tức là " HTTPS = HTTP + SSL " |
| **Được phát triển bởi** | HTTP được phát triển bởi ' Tim Berners-Lee'. | HTTPS được phát triển bởi ' Netscape Communications '. |
| **Sử dụng** | Giao thức này được sử dụng với các trang web mà không cần bảo mật . | Giao thức này được sử dụng với các trang web cần thiết để được an toàn và đáng tin cậy trong quá trình chuyển đổi. |
| **Nơi sử dụng** | HTTP được sử dụng với trang blog, trang web thông tin, v.v. | HTTPS được sử dụng với các trang web mua sắm, các trang web ngân hàng, trang web xã hội, v.v. |
| **Cổng sử dụng** | HTTP sử dụng cổng số: 80 | HTTPS sử dụng cổng số: 443 |
| **Mã hóa** | HTTP không sử dụng mã hóa vì vậy ít an toàn hơn. | HTTPS sử dụng mã hóa để chuyển đổi an toàn để đảm bảo an toàn hơn. |
| **Chứng nhận** | HTTP không cần xác nhận. | HTTPS cần chứng nhận . |
| **Hoạt động tại** | Hoạt động ở "Application Layer " trong mô hình OSI . | Hoạt động ở " Transport Layer " trong mô hình OSI. |
| **URL** | URL HTTP bắt đầu bằng "http: // " | URL HTTPS bắt đầu với " https: // " |
| **Chu trình hành động** | Trình duyệt sẽ mở kết nối TCP (Transmission Control Protocol) sau đó gửi yêu cầu HTTP đến máy chủ, máy chủ phản hồi thông qua HTTP đến trình duyệt và sau đó kết nối TCP bị đóng. | Đầu tiên xác thực được thực hiện sau đó lựa chọn các mật mãhoặc các thuật toán mật mã mà khách hàng và máy chủ hỗ trợ sau đó kỹ thuật mã hóa được sử dụng và sau đó tạo ra kết nối SSL(Secure Socket Layer) mã hóa và sau đó yêu cầu HTTP sẽ làm việc để tiếp tục ... |

* Sự khác biệt lớn nhất giữa HTTP và HTTPS là chứng chỉ SSL. Về cơ bản, HTTPS là một giao thức HTTP với bảo mật bổ sung. Tuy nhiên, trong thời đại mà mọi thông tin đều được số hóa, thì giao thức HTTPS lại trở nên cực kỳ cần thiết cho bảo mật website. Dù bạn sử dụng máy tính cá nhân hay công cộng, các tiêu chuẩn SSL sẽ luôn đảm bảo liên lạc giữa máy khách và máy chủ được an toàn, chống bị dòm ngó.
* **Port trên HTTP và HTTPS**

Định nghĩa đơn giản Port chính là một cổng để xác định thông tin nhận được trên máy khách sau đó phân loại gửi đến máy chủ. Mỗi một Port có số hiệu riêng với chức năng riêng biệt. Ví dụ như để gửi và nhận email được thực hiện  qua Port 25, giao thức truyền tải file thực hiện qua Port 21. Còn HTTP sử dụng Port 80 trong khi HTTPS là Port 443.

* **Mã hóa trên HTTP và HTTPS**

Như đã nói ở trên khi HTTPS được mã hóa thông tin, sử dụng SSL/ TSL tiêu chuẩn công nghệ bảo mật, truyền thông mã hóa giữa máy chủ Web server và trình duyệt. Với HTTP thì hoàn toàn không.

* **Mức độ bảo mật HTTP vs HTTPS**

HTTPS hỗ trợ việc xác thực tính đích danh của website mà máy khách truy cập thông qua việc kiểm tra xác thực bảo mật (Security Certificate). Các xác thực bảo mật này được cung cấp và xác minh bởi các CA (Certificate Authority) uy tín. Khi được xác thực từ CA người dùng sẽ biết được mình đang truy cập vào đúng website cần tìm thay vì một web mạo danh nào đó. Việc bảo mật HTTPS không phải là 100% an toàn nhưng tốt hơn HTTP rất nhiều. Tất nhiên với HTTP không được mã hóa thông tin nên rất dễ bị Hacker tấn công.

* HTTPS hiển nhiên là an toàn hơn so với HTTP rất nhiều trong việc mã hóa dữ liệu, bảo mật thông tin cá nhân. Tuy nhiên, ưu điểm của HTTP là tốc độ phản hồi của website truy cập nhanh hơn HTTPS rất nhiều và được sử dụng cho các trang tin tức cần thông tin nhanh, còn phải nhập dữ liệu như tài khoản ngân hàng, email cá nhân thì nên sử dụng HTTPS. Ngoài ra chúng ta cũng dễ dàng nhận biết với biểu tượng khóa ở thanh địa chỉ để phân biệt website đó có sử dụng HTTPS hay không.

Tóm lại, HTTP và HTTPS đơn giản đều là các giao thức để truyền dữ liệu từ Web Server đến trình duyệt web của nguồi truy cấp và ngược lại. Tuy nhiên hiện nay Chrome ưu tiên cho các trang web sử dụng giao thức HTTPS vì mang tính bảo mật cao và cảnh báo đối với các trang web sử dụng giao thức HTTP.

## 2.Tìm hiểu một ứng dụng trong thực tế và phân tích kiến trúc chương trình của nó

## XÂY DỰNG ỨNG DỤNG CLIENT/SERVER NHẬN HÌNH ẢNH TỪ CAMERA GỬI VỀ MÁY TÍNH

Method gửi ảnh từ Camera lên Server.

// Gửi ảnh từ camera lên Server

private void GuiDl()

{

while (true)

{

try

{

UdpClient send = new UdpClient();

IPEndPoint iepRemote = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 6969);

byte[] data1=new byte[1024];

string camera = "ca1";

byte[] data = new byte[1024];

data = Encoding.ASCII.GetBytes(camera);

if (anh != null)

{

try

{

Image tam = ResizeByWidth(anh, 100);

data1 = imageToByteArray(tam);

byte[] data2 = Combine(data, data1);

client.send(data2);

data2 = null;

send.Close();

}catch (Exception e) {}

}

}catch (Exception e) {}

Thread.Sleep(sleep);

}

}

Method cắt ảnh.

public Image ResizeByWidth(Image img, int width)

{

// lấy chiều rộng và chiều cao ban đầu của ảnh

int originalW = img.Width;

int originalH = img.Height;

// lấy chiều rộng và chiều cao mới tương ứng với chiều rộng truyền vào của ảnh (nó sẽ giúp ảnh của chúng ta sau khi resize vần giứ được độ cân đối của tấm ảnh

int resizedW = width;

int resizedH = (originalH \* resizedW) / originalW;

// tạo một Bitmap có kích thước tương ứng với chiều rộng và chiều cao mới

Bitmap bmp = new Bitmap(resizedW, resizedH);

// tạo mới một đối tượng từ Bitmap

Graphics graphic = Graphics.FromImage((Image)bmp);

graphic.InterpolationMode = InterpolationMode.High;

// vẽ lại ảnh với kích thước mới

graphic.DrawImage(img, 0, 0, resizedW, resizedH);

// gải phóng resource cho đối tượng graphic

graphic.Dispose();

// trả về anh sau khi đã resize

return (Image)bmp;

}

Method chuyển ảnh thành byte

//Chuyển ảnh thành byte

private byte[] imageToByteArray(Image imageIn)

{

MemoryStream ms = new MemoryStream();

imageIn.Save(ms, System.Drawing.Imaging.ImageFormat.Png);

return ms.ToArray();

}

Server nhận và gửi dữ liệu.

String loi = "loi";

String kiemtra = "";

String[] chuoi = null;

Program p = new Program();

//Nhận dữ liệu từ Client gửi lên

UdpClient receiver = new UdpClient(6969);

IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

byte[] data = new byte[1024];

data = receiver.Receive(ref iep);

kiemtra = Encoding.ASCII.GetString(data);

String moi = kiemtra;

kiemtra = kiemtra.Substring(0, 3);

Console.WriteLine("nhan duoc du lieu tu Client {0}", kiemtra );

if (kiemtra.Equals("log"))

{

//XỬ lý chuỗi khi đăng nhập

chuoi = p.Cahtchuoi(data);

//Kiểm tra CSDL

data = p.Kiemtracsdl(chuoi[0], chuoi[1]);

}

//Gửi dữ liệu về Client

iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9696);

receiver.Send(data, data.Length, iep);

kiemtra = null;

while (true)

{

try

{

//Nhận dữ liệu từ ClientCamera gửi lên

data = receiver.Receive(ref iep);

kiemtra = Encoding.ASCII.GetString(data);

kiemtra = kiemtra.Substring(0, 3);

Console.WriteLine("nhan duoc du lieu tu Client {0}", data.Length);

if (kiemtra.Equals("log"))

{

//XỬ lý chuổi khi đăng nhập

chuoi = p.Cahtchuoi(data);

//Kiểm tra CSDL

data = p.Kiemtracsdl(chuoi[0], chuoi[1]);

}

//Gửi dữ liệu về ClientPhone

iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9696);

receiver.Send(data, data.Length, iep);

kiemtra = null;

}

catch (Exception e) { }

}

Method nhận dữ liệu từ Server gửi đến

//Nhận dữ liệu từ Server gửi đến

private void NhanDl()

{

while (true)

{

try

{

byte[] tencam = null;

byte[] data = clPhone.Receive(Port);

//Chuyển byte thành chuỗi

String kiemtra = Encoding.ASCII.GetString(data);

int m = data.Length;

//Lấy tên Camera

tencam = data.Take(3).ToArray();

//Lấy dữ liệu ảnh

data = data.Skip(3).Take(m - 3).ToArray();

String ten = Encoding.ASCII.GetString(tencam);

string kiemtra1 = kiemtra.Substring(0, 3);

String chuoi = kiemtra.Substring(3);

byte[] data1 = Encoding.ASCII.GetBytes(chuoi);

//Kiểm tra dữ liệu nhận về

if (!(kiemtra1.Equals("tat"))&&ten.Equals("ca1"))

{

//Hiển thị ảnh lên PictureBox

pictureBox1.Image = byteArrayToImage(data);

}

}

catch (Exception e) { }

}

}

Method chuyển byte thành ảnh

private Image byteArrayToImage(byte[] byteArrayIn)

{

Image returnImage = null;

try

{

MemoryStream ms = new MemoryStream(byteArrayIn);

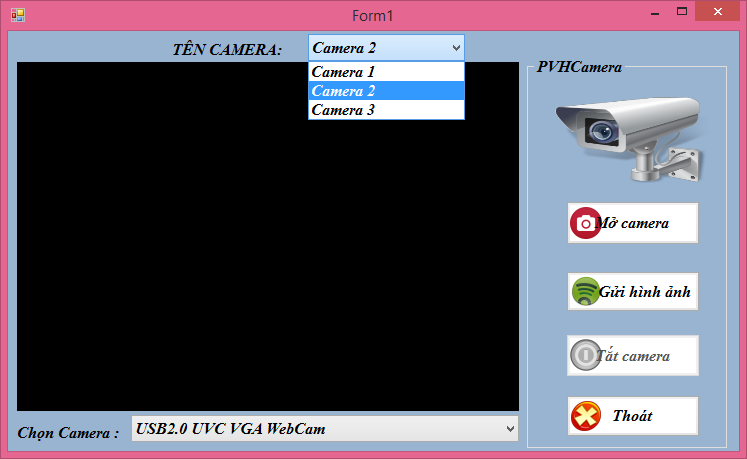
returnImage = Image.FromStream(ms);

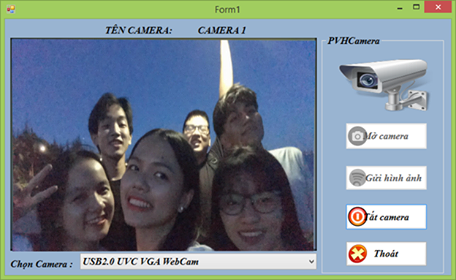
}

catch (Exception e) { }

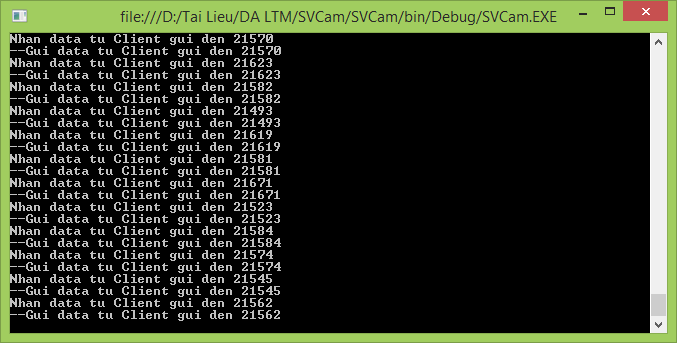
return returnImage;

Client 1:





### SERVER



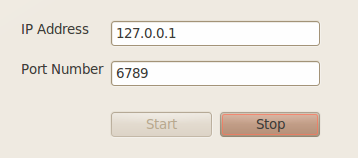
### CLIENT 2



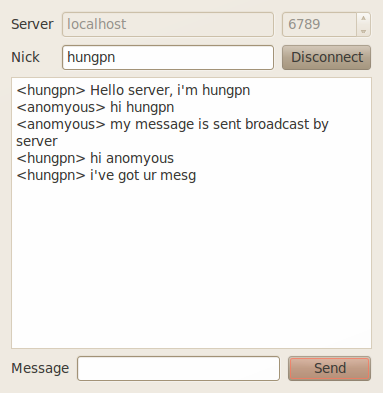
**XÂY DỰNG ỨNG DỤNG QT CHATROOM ĐƠN GIẢN**

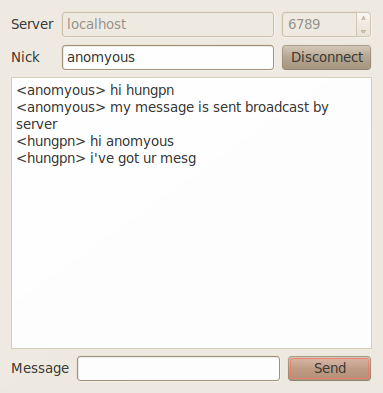
- **Ứng dụng gồm 2 phần**:

+ Chương trình **ChatServer**: Quản lý các client kết nối đến. Nhận các thông điệp và gửi broadcast lại các thông điệp này đến tất cả các client.

[](https://sites.google.com/site/embedded247/npcourse/xay-dung-ung-dung-qt-chatroom-don-gian/chatserver.png?attredirects=0)

+ Chương trình **ChatClient**: Kết nối đến Server, gửi các thông điệp đến server và nhận lại các thông điệp được server broadcast lại.

[](https://sites.google.com/site/embedded247/npcourse/xay-dung-ung-dung-qt-chatroom-don-gian/chatClient.png?attredirects=0)

[](https://sites.google.com/site/embedded247/npcourse/xay-dung-ung-dung-qt-chatroom-don-gian/chatClient2.png?attredirects=0)

* **Xây dựng chương trình ChatServer**

Mã nguồn chatserver.h

#ifndef CHATSERVER\_H

#define CHATSERVER\_H

#include <QWidget>

#include <QTcpServer>

#include <QList>

#include <QHash>

class QBuffer;

class QTcpSocket;

namespace Ui {

    class ChatServer;

}

class ChatServer : public QWidget

{

    Q\_OBJECT

public:

    explicit ChatServer(QWidget \*parent = 0);

    ~ChatServer();

private:

    QTcpServer \*server;

    //Danh sach chua cac connection de phuc vu client ket noi den

    QList<QTcpSocket\*> connections;

    //Mang bo dem du lieu, moi phan tu tuong ung du lieu cua mot socket

    QHash<QTcpSocket\*, QBuffer\*> buffers;

    Ui::ChatServer \*ui;

private slots: //Tao slots tu dong

    void on\_btnStop\_clicked();

    void on\_btnStart\_clicked();

private slots: //Tao bang cach viet ma

    void addConnection(); //Xu ly khi co mot client ket noi den

    void removeConnection(); //Xu ly khi co mot client ngat ket noi

    void receiveMessage(); //Nhan thong diep tu cac client

};

#endif // CHATSERVER\_H

**Mã nguồn chatserver.cpp**

 #include "chatserver.h"

#include "ui\_chatserver.h"

#include <QHostAddress>

#include <QTcpSocket>

#include <QBuffer>

static const int DEFAULT\_PORT = 6789;

int port;

QHostAddress server\_addr;

ChatServer::ChatServer(QWidget \*parent) :

    QWidget(parent),

    ui(new Ui::ChatServer)

{

    ui->setupUi(this);

    server = new QTcpServer();

    //Dang ky signal/slot cho su kien co client ket noi den

    connect(server, SIGNAL(newConnection()), this, SLOT(addConnection()));

    server\_addr = QHostAddress::LocalHost; //Lay dia chi may host

    //server\_addr = QHostAddress::Any;

    port = DEFAULT\_PORT; //Cong mac dinh

    ui->lineEditIP->setText(server\_addr.toString()); //Hien dia chi nay len lineEdit

    ui->lineEditIP->isEnabled();

    ui->lineEditPort->setText(QString::number(port));

}

ChatServer::~ChatServer()

{

    delete ui;

}

void ChatServer::on\_btnStart\_clicked()

{

    port = ui->lineEditPort->text().toInt();

    //server bat dau lang nghe ket noi

    bool b = server->listen(server\_addr, port);

    if(b)

    {

        qDebug("server started");

        ui->btnStart->setEnabled(false);

        ui->btnStop->setEnabled(true);

    }

    else

    {

        qDebug("server can't start! check IP or Port");

    }

}

void ChatServer::on\_btnStop\_clicked()

{

    if(server->isListening()) //Neu server dang lang nghe

    {

        server->close(); //thi dung

        ui->btnStart->setEnabled(true);

        ui->btnStop->setEnabled(false);

    }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ham xu ly su kien khi co client ket noi den

Thuc hien:

- Tao socket phuc vu ket noi va them vao danh sach quan ly

- Tao bo dem du lieu tuong ung cho socket do va them vao danh sach quan ly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ChatServer::addConnection()

{

    QTcpSocket\* connection = server->nextPendingConnection();

    connections.append(connection); //Them ket noi vao danh sach

    QBuffer \*buffer = new QBuffer(this); //Tao bo dem du lieu cho connection nay

    buffer->open(QIODevice::ReadWrite);

    buffers.insert(connection, buffer); //Luu vao danh sach

    //Dang ky signal/slot cho moi connection khi co su kien disconnect hoac du lieu gui toi

    connect(connection, SIGNAL(disconnected()), this, SLOT(removeConnection()));

    connect(connection, SIGNAL(readyRead()), this, SLOT(receiveMessage()));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ham xu ly su kien khi co client ngat ket noi

Thuc hien:

- Lay socket cua ket noi can ngat

- Lay bo dem du lieu dang quan ly

- Thuc hien ngat ket noi va giai phong bo dem du lieu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ChatServer::removeConnection()

{

    //Lay socket gui su kien ngat ket noi

    QTcpSocket\* socket = static\_cast<QTcpSocket\*>(sender());

    QBuffer \*buffer = buffers.take(socket); //Lay bo dem du lieu cua ket noi nay (tu danh sach bo dem)

    buffer->close(); //Ket thuc gui nhan

    buffer->deleteLater();//giai phong bo dem

    connections.removeAll(socket); //Xoa socket nay khoi danh sach ket noi server dang quan ly

    socket->deleteLater(); //Giai phong socket

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ham xu ly su kien khi co du lieu san sang nhan tu mot socket

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ChatServer::receiveMessage()

{

    //Xac dinh ket noi nao co thong diep den

    QTcpSocket\* socket = static\_cast<QTcpSocket\*>(sender());

    //Lay bo dem du lieu tuong ung voi socket (tu bang hash buffers)

    QBuffer \*buffer = buffers.value(socket);

    //ghi toan bo du lieu cua socket nay vao bo dem (buffer) da xac dinh o tren

    qint64 bytes = buffer->write(socket->readAll()); //bytes chua kich thuoc ghi thanh cong

    //dich chuyen ve dau bo dem bang ham seek

    buffer->seek(buffer->pos() - bytes);

    //Vong lap doc tung dong (line) cua thong diep chua trong bo dem

    while (buffer->canReadLine())

    {

            QByteArray line = buffer->readLine();//doc tung dong

            //Gui broadcast den tat ca cac ket noi dang quan ly

            foreach (QTcpSocket\* connection, connections)

            {

                    connection->write(line); //Gui bang cach ghi ra socket

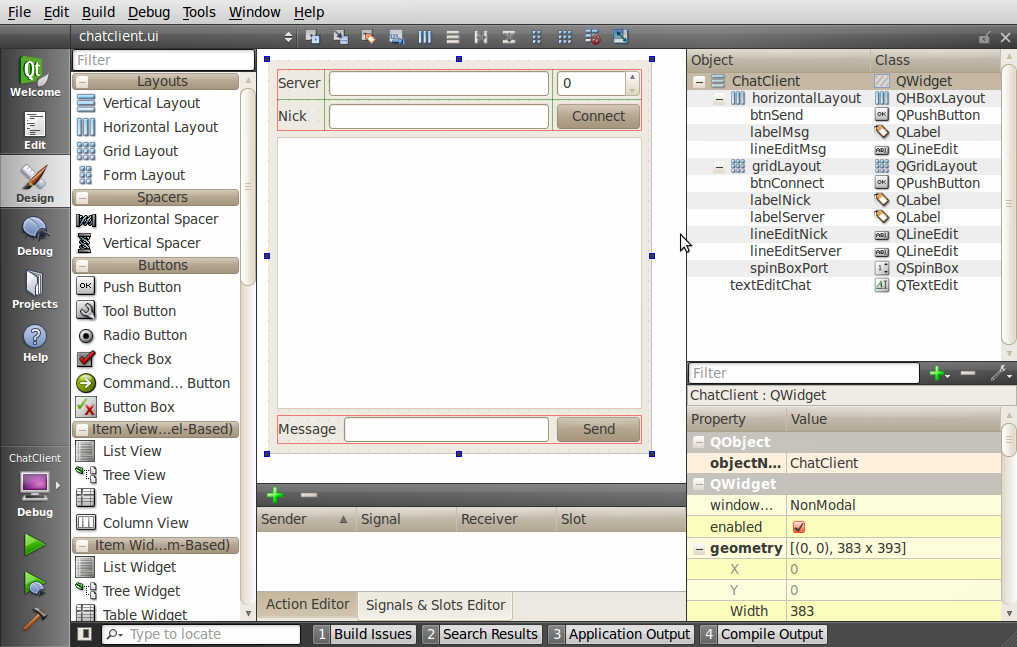
            }

    }

}

**- Xây dựng chương trình ChatClient**

Thiết kế giao diện như hình minh họa:

[](https://sites.google.com/site/embedded247/npcourse/xay-dung-ung-dung-qt-chatroom-don-gian/clientdesign.png?attredirects=0)

**Sử dụng kỹ thuật layout:**

       grid layout cho nhóm điều khiển ở trên cùng (labelServer, labelNick, lineEditServer, lineEditNick, spinBoxPort, btnConnect)

       horizontal layout cho nhóm điều khiển ở dưới cùng (labelMsg, lineEditMsg, btnSend)

       Sau đó, sử dụng vertical layout cho cả form (gồm 2 nhóm trên và điều khiển textEditChat ở giữa).

**+ Mã nguồn chatclient.h**

|  |
| --- |
| #ifndef CHATCLIENT\_H  #define CHATCLIENT\_H  #include <QWidget>  #include <QBuffer>  #include <QTcpSocket>  namespace Ui {      class ChatClient;  }  class ChatClient : public QWidget  {      Q\_OBJECT  public:      explicit ChatClient(QWidget \*parent = 0);      ~ChatClient();  //Khai bao cac ham slot  private slots:      void setConnected();      void setDisconnected();      void toggleConnection();      void sendMessage();      void receiveMessage();  private:      Ui::ChatClient \*ui;      QBuffer \*buffer;      QTcpSocket \*socket;  };  #endif // CHATCLIENT\_H |

**+ Mã nguồn chatclient.cpp**

|  |
| --- |
| #include "chatclient.h"  #include "ui\_chatclient.h"  static const int DEFAULT\_PORT = 6789;  ChatClient::ChatClient(QWidget \*parent) :      QWidget(parent),      ui(new Ui::ChatClient)  {      ui->setupUi(this);      //Khoi tao cho cac dieu khien (widget)      ui->spinBoxPort->setRange(1000,32767);      ui->spinBoxPort->setValue(DEFAULT\_PORT);      ui->lineEditServer->setText("localhost");      ui->lineEditNick->setText("hungpn");      //Khai bao socket va buffer du lieu      socket = new QTcpSocket(this);      buffer = new QBuffer(this);      buffer->open(QIODevice::ReadWrite);      //Dang ky signal/slot cho cac su kien      //Su kien click nut Connect      connect(ui->btnConnect, SIGNAL(clicked()), this, SLOT(toggleConnection()));      //Su kien go message va enter (gui di)      connect(ui->lineEditMsg, SIGNAL(returnPressed()), this, SLOT(sendMessage()));      //Hoac click nut Send (gui di)      connect(ui->btnSend, SIGNAL(clicked()), this, SLOT(sendMessage()));      //Su kien ket noi Socket thanh cong      connect(socket, SIGNAL(connected()), this, SLOT(setConnected()));      //Su kien ngat ket noi      connect(socket, SIGNAL(disconnected()), this, SLOT(setDisconnected()));      //Su kien san sang nhan du lieu      connect(socket, SIGNAL(readyRead()), this, SLOT(receiveMessage()));      //Ban dau chua ket noi      setDisconnected();  }  ChatClient::~ChatClient()  {      delete ui;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    Ham xu ly cac dieu khien khi ket noi thanh cong  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void ChatClient::setConnected()  {      ui->lineEditServer->setEnabled(false);      ui->spinBoxPort->setEnabled(false);      ui->lineEditNick->setEnabled(true);      ui->lineEditMsg->setEnabled(true);      ui->textEditChat->setEnabled(true);      ui->textEditChat->clear();      ui->btnSend->setEnabled(true);      ui->btnConnect->setText("Disconnect");  }  void ChatClient::setDisconnected()  {      ui->lineEditServer->setEnabled(true);      ui->spinBoxPort->setEnabled(true);      ui->lineEditNick->setEnabled(false);      ui->lineEditMsg->setEnabled(false);      ui->textEditChat->setEnabled(false);      ui->textEditChat->clear();      ui->btnSend->setEnabled(false);      ui->btnConnect->setText("Connect");  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    Ham tat/bat ket noi den server  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void ChatClient::toggleConnection()  {      if(socket->state() == QAbstractSocket::UnconnectedState)      {          socket->connectToHost(ui->lineEditServer->text(), ui->spinBoxPort->value());      }      else      {          socket->disconnectFromHost();      }  }  void ChatClient::sendMessage()  {      // "<nick> message\n"      QString nick = ui->lineEditNick->text().toLatin1();      QString msg = ui->lineEditMsg->text().toLatin1();      //socket->write("<" + nick + "> " + msg + "\n");      socket->write("<" + ui->lineEditNick->text().toLatin1() + "> " + ui->lineEditMsg->text().toLatin1() + "\n");      ui->lineEditMsg->clear();  }  void ChatClient::receiveMessage()  {      // missing some checks for returns values for the sake of simplicity      qint64 bytes = buffer->write(socket->readAll());      // go back as many bytes as we just wrote so that it can be read      buffer->seek(buffer->pos() - bytes);      // read only full lines, line by line      while (buffer->canReadLine())      {              QString line = buffer->readLine();              ui->textEditChat->append(line.simplified());      }  } |

**- Một số lớp thư viện Qt sử dụng trong ứng dụng**

Chương trình ChatServer có sử dụng một số lớp thư viện Qt quan trọng để hỗ trợ xử lý là:

        Lớp QList để quản lý một danh sách phần tử

        Lớp QBuffer cung cấp bộ đệm dữ liệu

        Lớp QHash quản lý một từ điển (dictionary) dựa trên bảng hash

**+ Lớp Qlist:**

|  |
| --- |
| QList<QTcpSocket\*> connections; |

Tạo ra một danh sách connections để quản lý các kết nối từ client đến. Một kết nối mới (là một socket) sẽ được thêm vào danh sách này khi có sự kiện

newConnection và sẽ được xóa khỏi danh sách khi có sự kiện ngắt kết nối (disconnected)

//Thêm một kết nối mới

|  |
| --- |
| QTcpSocket\* connection = server->nextPendingConnection();      connections.append(connection); //Them ket noi vao danh sach |

 //Xóa một kết nối khỏi danh sách khi bị ngắt

|  |
| --- |
| QTcpSocket\* socket = static\_cast<QTcpSocket\*>(sender());  //...  connections.removeAll(socket); |

**+ Lớp**[**QHash**](http://doc.qt.nokia.com/4.7-snapshot/qhash.html)

Lớp QHash cung cấp một từ điển dựa trên bảng hash.

Qhash<Key, T> là một lớp chứa cặp (key, value) và cung cấp cơ chế để truy cập nhanh vào value (giá trị) liên kết với một key (khóa)

Khai báo:

|  |
| --- |
| QHash<QTcpSocket\*, QBuffer\*> buffers; |

Tạo ra một bảng hash có tên buffers để quản lý các cặp <socket, buffer>

Trong đó: socket là kết nối được tạo ra cho mỗi client kết nối đến server và buffer là bộ đệm (đối tượng lớp Qbuffer) để chứa dữ liệu gửi/nhận trên socket đó.

 Câu lệnh:

|  |
| --- |
| buffers.insert(connection, buffer); //Luu vao danh sach |

Sẽ chèn thêm một cặp <connection, buffer> vào bảng hash buffers để quản lý các kết nối và dữ liệu của nó.

Để lấy bộ đệm dữ liệu ứng với một kết nối (socket):

|  |
| --- |
| //Lay bo dem du lieu cua socket nay      QBuffer \*buffer = buffers.value(socket); |

**+ Lớp**[**QBuffer**](http://doc.qt.nokia.com/4.7-snapshot/qbuffer.html)

Lớp này cung cấp cơ chế để giao tiếp với một mảng dữ liệu kiểu byte (**QByteArray**) sử dụng giao diện QIODevice

Trong hàm ChatServer::receiveMessage() xử lý sự kiện nhận thông điệp từ Client

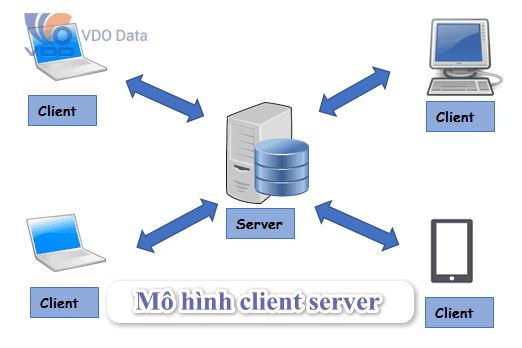
|  |
| --- |
| //Xac dinh ket noi nao co thong diep den      QTcpSocket\* socket = static\_cast<QTcpSocket\*>(sender());      //Lay bo dem du lieu tuong ung voi socket (tu bang hash buffers)      QBuffer \*buffer = buffers.value(socket);      //ghi toan bo du lieu cua socket nay vao bo dem (buffer) da xac dinh o tren      qint64 bytes = buffer->write(socket->readAll()); //bytes chua kich thuoc ghi thanh cong      //dich chuyen ve dau bo dem bang ham seek      buffer->seek(buffer->pos() - bytes);      //Vong lap doc tung dong (line) cua thong diep chua trong bo dem      while (buffer->canReadLine())      {              QByteArray line = buffer->readLine();//doc tung dong              //Gui broadcast den tat ca cac ket noi dang quan ly              foreach (QTcpSocket\* connection, connections)              {                      connection->write(line); //Gui bang cach ghi ra socket              }      } |

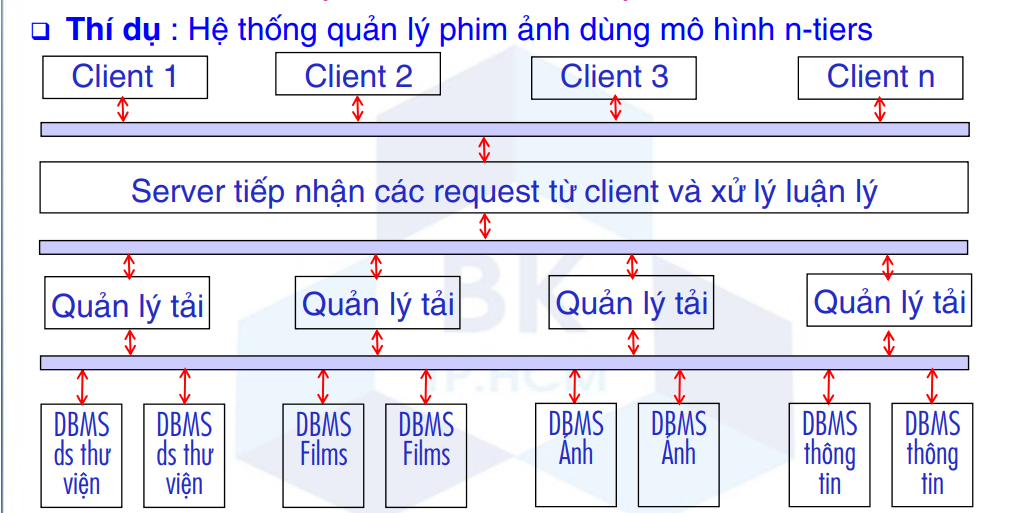
3.Chọn 1 kiến trúc chương trình và thiết kế ứng dụng theo kiến trúc đã chọn

* Kiến trúc này còn được biết đến với tên kiến trúc **Client-Server**. Kiến trúc này gồm 2 chương trình thực thi: chương trình Client và chương trình Server. Cả hai chương trình có thể được thực thi trên cùng một máy tính hay trên hai máy tính khác nhau.
* Kiến trúc client-server (client-server Architecture) đặc tả : Hệ thống gồm 2 loại phần tử chức năng : server cung cấp 1 số dịch vụ, client là phần tử sử dụng dịch vụ bằng cách truy xuất đến server tương ứng.

Client 🡪 Server

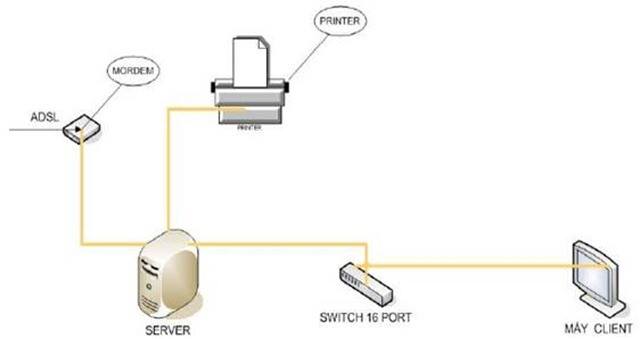
* Kiến trúc làm việc dựa trên mô hình request-response. Client gửi request đến server để lấy thông tin và server trả dữ liệu về.
* Client và Server trao đổi thông tin với nhau dưới dạng các thông điệp (Message). Thông điệp gởi từ Client sang Server gọi là các thông điệp  yêu cầu (Request Message) mô tả công việc mà phần Client muốn Server thực hiện.
* **Fat Client**: Các máy trạm trong mô hình này gọi là máy khách (client). Đây là nơi gởi các yêu cầu xử lý về máy chủ (server).Phần phía Client tổ chức giao tiếp với người dùng, với môi trường bên ngoài tại trạm làm việc và với phía Server. Sau khi tiếp nhận yêu cầu của người dùng, phần phía client thành lập các query string gửi về phía Server, tiếp nhận kết quả và tổ chức trình diễn chúng.
* **Fat Server**: Máy chủ xử lý và gửi kết quả về máy khách. Máy khách có thể tiếp tục xử lý các kết quả này để phục vụ cho công việc.Phần phía Server quản lý các giao tiếp môi trường bên ngoài tại Server và với các Client, tiếp nhận các yêu cầu dưới dạng các xâu ký tự (query string). Sau khi phân tích các query string, phần phía Server sẽ xử lý dữ liệu và gửi kết quả trả lời về phía các Client.
* Mọi website bạn truy cập như blog, Facebook, Twitter đều được xây dựng trên kiến trúc client-server.





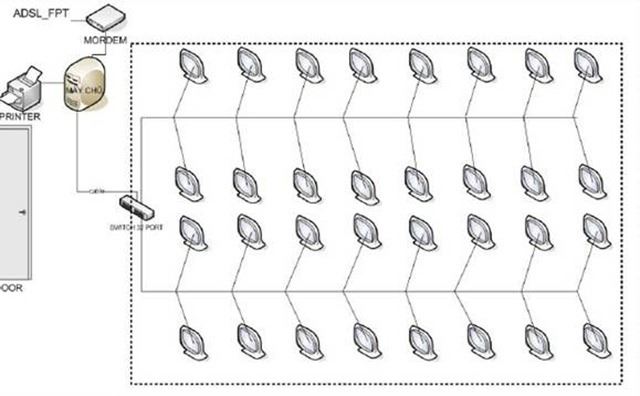
**Thiết kế phòng máy Internet bằng mạng LAN (Client-Server)**

1. **Sơ đồ logic cho phòng máy**



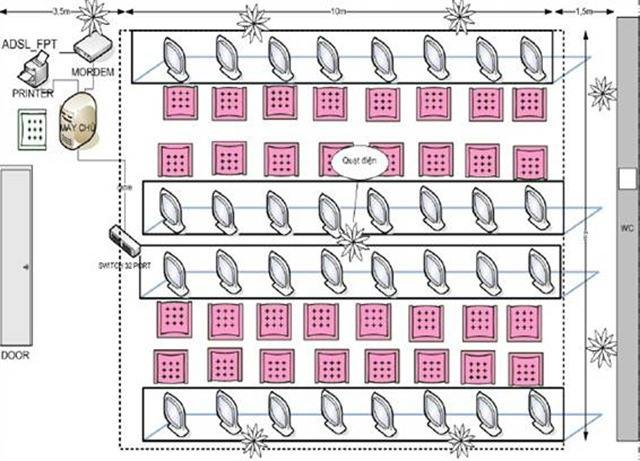
Hình 1: Sơ đồ logic cho phòng máy Internet giữa các máy Client và máy Server

1. **Sơ đồ vật lý cho phòng máy**



Hình 1: Sơ đồ vật lí cho phòng máy Internet giữa các máy Client và máy Server

1. **Sơ đồ thiết kế chi tiết phòng máy**



Hình 1: Sơ đồ logic cho phòng máy Internet với các trang thiết bị

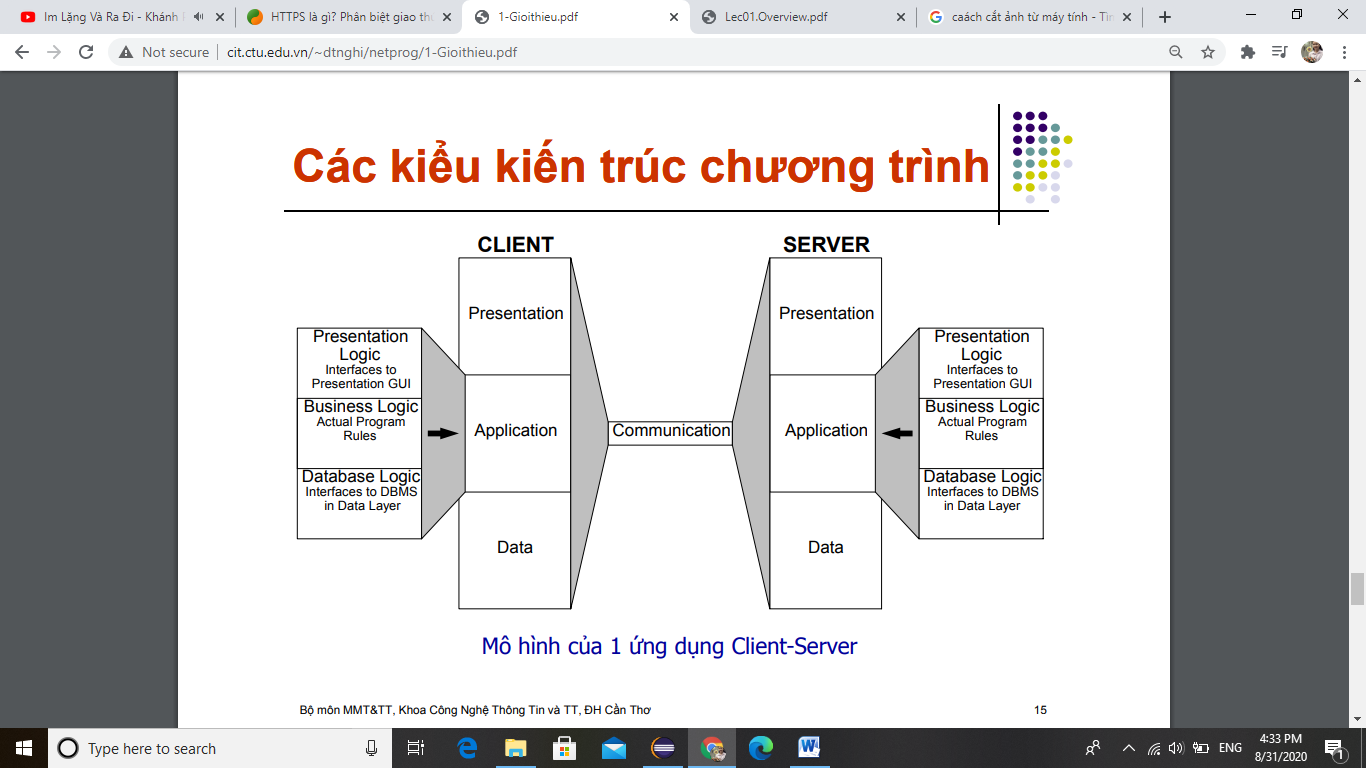
* **Phần mềm máy trạm (Client)**

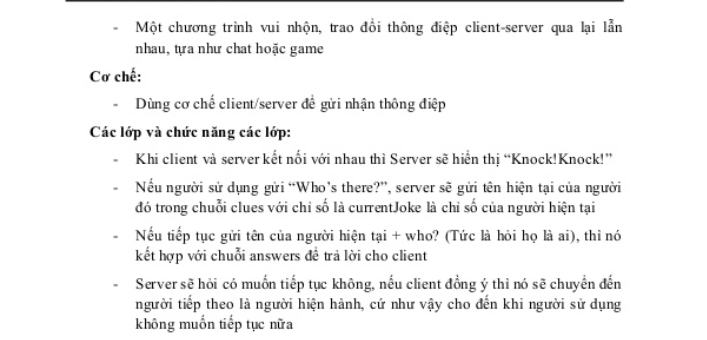
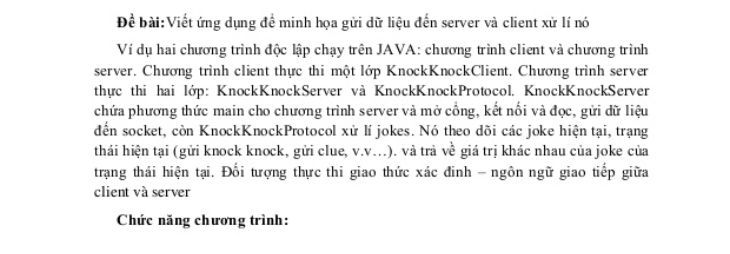
Sử dụng hệ điều hành Microsoft Windows XP Professional SP2 (có sẵn serial key khi mua đĩa CD nên không lo lắng về chi phí bản quyền hệ điều hành này). Dùng chung cho hệ thống công ty hoặc phòng máy.

* Chương trình Microsoft Office Edition 2003 English (cũng đã có sẵn serial key).
* Chương trình Internet Download Manager trị giá 399.000 VNĐ
* Chương trình Yahoo Messenger 10 (Free)
* Các game online tải từ trang chủ của game hoặc game offline
* Chương trình duyệt web Internet Explorer 8.0 (có thể nâng cấp lên từ version 7 của Windows XP SP2).
* Chương trình xem phim nghe nhạc Windows Media Player 11 (có thể nâng cấp lên từ version 10 của Windows XP SP2)
* Chương trình Vietkey (Free)
* Chương trình đọc file .pdf Microsoft Acrobat Reader (Free)
* Chương trình đóng băng ổ cứng Deepfreeze Enterprise (Free)
* **Chương trình máy chủ (Server)**

Sử dụng hệ điều hành Microsoft Windows Server 2003 Standard Edition trị giá 550.000 VNĐ (có thể liên lạc với một số cá nhân kinh doanh phần mềm)

* Chương trình Microsoft Office Edition 2003 English (cũng đã có sẵn serial key).
* Chương trình duyệt web Internet Explorer 8.0 (có thể nâng cấp lên)
* Chương trình xem phim nghe nhạc Windows Media Player 11 (có thể nâng cấp)
* Chương trình Vietkey (Free)
* Chương trình đọc file .pdf Microsoft Acrobat Reader (Free)
* Chương trình đóng băng ổ cứng Deepfreeze Enterprise (Free)
* Chương trình quản lý phòng máy Nescafe hoặc được cung cấp bởi nhà phân phối mạng Internet khi yêu cầu cho phòng máy tính.
* Hình mô tả:





**TÀI LIỆU THAM KHẢO**

1. *“*[*https://tatthanh.com.vn/https-la-gi-phan-biet-giao-thuc-http-va-https*](https://tatthanh.com.vn/https-la-gi-phan-biet-giao-thuc-http-va-https)*”* : Phân biệt giao thức HTTP và HTTPS.
2. *“*<https://sites.google.com/site/embedded247/npcourse/tim-hieu-ky-thuat-video-streaming>*”:* Tìm hiểu kĩ thuật video streaming

*3.* “<https://gpcoder.com/3679-xay-dung-ung-dung-client-server-voi-socket-trong-java/>”: Xây dựng ứng dụng Client-Server với Socket trong Java.