**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VIỆT - HÀN**



**BÁO CÁO ĐỒ ÁN CƠ SỞ (4)**

**ĐỀ TÀI:**

**Tìm Hiểu Giao Thức VNC. Xây Dựng Hệ Thống Giám Sát Và Điều Khiển Từ Xa.**

Sinh viên thực hiện : **BÙI VĂN VẠN QUÝ**

**TRƯƠNG THÀNH QUÝ**

Giảng viên hướng dẫn : **TS.NGUYỄN QUANG VŨ**

Lớp : **18IT5**

***Đà nẵng, tháng 01 năm 2021***

**LỜI CẢM ƠN**

Nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Quang Vũ – giảng viên hướng dẫn môn đồ án cơ sở (4) đã trang bị cho chúng em những kiến thức, kỹ năng cơ bản cần có để hoàn thành đề tài này.

Tuy nhiên trong quá trình nghiên cứu đề tài, do kiến thức chuyên ngành còn hạn chế nên nhóm chúng em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tấm và góp ý của thầy/cô bộ môn để tài của em được đầy đủ và hoàn chỉnh hơn

Xin chân thành cảm ơn.

**NHẬN XÉT**

(Của cơ quan thực tập, nếu có)

Bold, size 16, xếp sau trang Lời cảm ơn

size 13**………………………………..**

**………………………………………………………………………………………**

**………………………………………………………………………………………**

**………………………………………………………………………………………**

**………………………………………………………………………………………**

**………………………………………………………………………………………**

**NHẬN XÉT**

**(Của giảng viên hướng dẫn)**

Mục Lục

[PHẦN I : GIỚI THIỆU 5](#_Toc59547768)

[1. Tổng quan : 5](#_Toc59547769)

[2. Mục đích, phương pháp, kết quả : 5](#_Toc59547770)

[PHẦN II : NỘI DUNG 6](#_Toc59547771)

[Chương 1 : Tổng Quan Về VNC. 6](#_Toc59547772)

[**1. VNC là gì ? 6**](#_Toc59547773)

[**2. Lịch sử hình thành : 7**](#_Toc59547774)

[**3. Từ nguyên : 7**](#_Toc59547775)

[**4. Hoạt động : 8**](#_Toc59547776)

[**5. Bảo mật : 9**](#_Toc59547777)

[**6. Giao thức RFP : 10**](#_Toc59547778)

[Chương 2 : Tổng Quan Về REAL.VNC 50](#_Toc59547779)

[**1. RealVNC là gì ? 50**](#_Toc59547780)

[**2. Lịch sử : 52**](#_Toc59547781)

[**3. Tính năng chính : 52**](#_Toc59547782)

[**4. Các chức năng chính của VNC Connect : 53**](#_Toc59547783)

[**5. Nền tảng , các phiên bản : 54**](#_Toc59547784)

[**6. Lý do chọn Real VNC ? 55**](#_Toc59547785)

# PHẦN I : GIỚI THIỆU

1. **Tổng quan :**

Điều khiển từ xa là một nhu cầu cần thiết hiện nay, đặc biệt với các trung tâm hỗ trợ từ xa thì không thể không triển khai các giải pháp điều khiển từ xa, giúp tiết kiệm được rất nhiều chi phí như: chi phí đi lại, thời gian đi lại, dễ dàng trao đổi, hướng dẫn, thao tác…vv.

Hiện nay có khá nhiều công nghệ để xây dựng mô hình điều khiển từ xa giữa các máy tính, điện thoại, vv... Trong đó có **VNC** (**Vitual Network Computing**) ,là một hệ thống [chia sẻ màn](https://en.wikipedia.org/wiki/Desktop_sharing) hình đồ họa sử dụng [giao thức **Bộ Đệm Khung Từ Xa (RFB)**](https://en.wikipedia.org/wiki/RFB_protocol) để điều khiển từ xa một [máy tính](https://en.wikipedia.org/wiki/Computer) khác . Nó truyền các sự kiện [bàn phím](https://en.wikipedia.org/wiki/Computer_keyboard) và [chuột](https://en.wikipedia.org/wiki/Computer_mouse) từ máy tính này sang máy tính khác, chuyển tiếp các cập nhật [màn hình](https://en.wikipedia.org/wiki/Computer_screen) đồ họa từ máy chủ sang máy khách thông qua [mạng](https://en.wikipedia.org/wiki/Computer_network) .

1. **Mục đích, phương pháp, kết quả :**

* **Mục đích :** 
  + Tìm hiểu về VNC, quá trình hình thành,lịch sử phát triển cũng như những giao thức, công nghệ sử dụng VNC
* Tìm hiểu về các phần mềm mã nguồn mở hỗ trợ truy cập từ xa sử dụng kỹ thuật VNC.
  + Xây dựng chương trình truy cập màn hình máy tính từ xa ứng dụng VNC
* **Phương pháp :**
* Đọc, dịch , nghiên cứu tài liệu .
* Tìm hiểu, nghiên cứu xây dựng hệ thống mô phỏng từ mã nguồn mở có sẵn .
* **Kết quả mong muốn :**
* Báo cáo tổng quan về VNC và các phần mềm mã nguồn mở hỗ trợ truy cập từ xa sử dụng kỹ thuật VNC.
* Xây dựng thành công "Chương trình truy cập màn hình máy tính từ xa ứng dụng VNC."

# PHẦN II : NỘI DUNG

## Chương 1 : Tổng Quan Về VNC :

1. **VNC là gì ?**

Trong lĩnh vực điện toán, mạng máy tính ảo ( **VNC** ) là một hệ thống [chia sẻ màn](https://en.wikipedia.org/wiki/Desktop_sharing) hình đồ họa từ xa sử dụng [giao thức **Remote Framebuffer** hay còn gọi là **(RFB)**](https://en.wikipedia.org/wiki/RFB_protocol) để điều khiển từ xa một [máy tính](https://en.wikipedia.org/wiki/Computer) khác. Giao thức mạnh mẽ này cho phép máy chủ cập nhật bộ đệm khung cho các máy tính từ xa .Nó truyền các sự kiện từ [bàn phím](https://en.wikipedia.org/wiki/Computer_keyboard) và [chuột](https://en.wikipedia.org/wiki/Computer_mouse) ở máy khách sang máy chủ, và chuyển tiếp các cập nhật [màn hình](https://en.wikipedia.org/wiki/Computer_screen) đồ họa theo hướng ngược lại qua [mạng](https://en.wikipedia.org/wiki/Computer_network).

VNC là nền tảng độc lập – hỗ trợ máy khách và máy chủ cho nhiều hệ điều hành dựa trên GUI và [Java](https://en.wikipedia.org/wiki/Java_Virtual_Machine) . Nhiều máy khách có thể kết nối với một máy chủ VNC cùng một lúc. Các ứng dụng phổ biến cho công nghệ này bao gồm hỗ trợ kỹ thuật từ xa và truy cập, thao tác, xử lí các tệp trên máy tính của người khác bằng máy tính của mình và có thể ngược lại.

VNC ban đầu được phát triển tại [phòng thí nghiệm nghiên cứu Olivetti & Oracle](https://en.wikipedia.org/wiki/Olivetti_%26_Oracle_Research_Lab) ở Cambridge, Vương quốc Anh. [Mã nguồn](https://en.wikipedia.org/wiki/Source_code) VNC [gốc](https://en.wikipedia.org/wiki/Source_code) là [mã nguồn mở](https://en.wikipedia.org/wiki/Open-source_software) theo [giấy phép của cộng đồng GNU](https://en.wikipedia.org/wiki/GNU_General_Public_License).

VNC sử dụng mô hình máy khách- máy chủ, nơi người xem được cài đặt trên máy tính cục bộ và kết nối từ xa với máy tính. Nó diễn giải cho người dùng các lệnh như nhấn phím hoặc nhấn chuột của người xem và thực hiện các thao tác tương ứng trên máy chủ. Tất cả thiết bị của VNC đều phải kết nối với TCP/IP và có cổng mở cho phép lưu lượng từ địa chỉ IP trên các thiết bị cần thiết.

VNC và RFB là [nhãn hiệu đã được đăng ký](https://en.wikipedia.org/wiki/Registered_trademark) của [RealVNC](https://en.wikipedia.org/wiki/RealVNC) Ltd. tại Mỹ và một số quốc gia khác.

Logo VNC.

1. **Lịch sử hình thành :**

Ở phòng thí nghiệm nghiên cứu của Olivetti & Oracletại Cambridge ở Anh đã phát triển VNC. Năm 1999, AT&T mua lại phòng thí nghiệm và năm 2002 đóng cửa các nỗ lực nghiên cứu của phòng thí nghiệm.

Các nhà phát triển đã làm việc nghiên cứu ra VNC khi còn ở phòng thí nghiệm AT&T bao gồm:

* Tristan Richardson (nhà phát minh)
* [Andy Harter](https://en.wikipedia.org/wiki/Andy_Harter) (trưởng dự án)
* [Quentin Stafford-Fraser](https://en.wikipedia.org/wiki/Quentin_Stafford-Fraser)
* James Weatherall
* [Andy Hopper](https://en.wikipedia.org/wiki/Andy_Hopper)

Sau khi đóng cửa phòng thí nghiệm vào năm 2002, một số thành viên của nhóm phát triển (bao gồm Richardson, Harter, Weatherall và Hopper) đã thành lập ra RealVNC, để tiếp tục công việc nghiên cứu phát triển VNC và sau đó lấy tên phần mềm là RealVNC.

Mã nguồn GPL ban đầu đã được đưa vào một số phiên bản khác của VNC. Việc phân [nhánh](https://en.wikipedia.org/wiki/Fork_(software_development)) như vậy đã không dẫn đến các vấn đề tương thích vì giao thức RFB được thiết kế theo hướng để mở rộng. Máy khách và máy chủ của VNC thương lượng khả năng của chúng bằng cách gửi các thông điệp cho nhau để sử dụng các tùy chọn thích hợp nhất được hỗ trợ ở cả hai đầu.

Kể từ năm 2013 , [RealVNC](https://en.wikipedia.org/wiki/RealVNC) Ltd tuyên bố thuật ngữ "VNC" là nhãn hiệu đã được đăng ký tại Hoa Kỳ và các quốc gia khác.

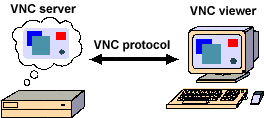
1. **Từ nguyên :**

Cái tên**Mạng máy tính ảo (VNC)** bắt nguồn từ công việc của ORL trên một máy khách có cấu hình tối thiểu gọi là Videotile và cũng có sử dụng giao thức RFB.

1. **Hoạt động :**

VNC hoạt động theo cơ chế client/server và sử dụng giao thức VNC. Giao thức VNC vô cùng đơn giản chỉ phục vụ cho một mục đích duy nhất đó là truy cập kênh giao diện đồ hoạ của máy tính người dùng ở xa qua mạng.

Giao thức VNC này được thiết kế trên ý tưởng của Remote Frame Buffer (RFB). VNC Client (Viewer) sẽ chia sẻ các input như (bàn phím, di chuyển chuột, click chuột,…) với VNC Server. VNC Server sẽ ghi lại các nội dung hiển thị Framebuffer và chia sẻ chúng lại cho VNC Client.



Như vậy thông thường mô hình client/server sẽ như sau.

* Máy tính cá nhân/máy chủ mà bạn muốn điều khiển sẽ cài đặt bản VNC (server+client).
* Máy khách chỉ cần chạy bản VNC Viewer (không cần cài đặt) là có thể điều khiển được.



**Ưu điểm của VNC**

* Kết nối máy tính từ xa, quản lý, xem theo dõi,…
* Kết nối được từ các phương tiện như laptop, điện thoại thông minh,..

**Nhược điểm VNC**

* Phụ thuộc đường truyền mạng.
* Ứng dụng có thể nặng nề.
* Hiệu suất thao tác đôi khi hơi chậm và không ổn định.
* Độ bảo mật không cao.

1. **Bảo mật :**

Theo mặc định, RFB không phải là một giao thức an toàn. Mặc dù [mật khẩu](https://en.wikipedia.org/wiki/Password) không được gửi ở dạng văn bản thuần túy (như trong [**Telnet**](https://en.wikipedia.org/wiki/Telnet) ), việc bẻ khóa có thể thành công nếu cả khóa [mã hóa](https://en.wikipedia.org/wiki/Encryption) và mật khẩu được mã hóa đều bị [đánh cắp](https://en.wikipedia.org/wiki/Packet_analyzer) từ người khác. Vì lý do này, chúng ta nên sử dụng mật khẩu có ít nhất 8 ký tự. Mặt khác cũng có giới hạn ít nhất 8 ký tự trên một số phiên bản của VNC. Nếu mật khẩu được gửi vượt quá 8 ký tự, các ký tự thừa sẽ bị loại bỏ và chuỗi bị cắt ngắn được so sánh với mật khẩu.

[UltraVNC](https://en.wikipedia.org/wiki/UltraVNC) có hỗ trợ sử dụng một Plugin mã hóa với mã nguồn mở, mã hóa toàn bộ phiên bản VNC bao gồm xác thực mật khẩu và truyền dữ liệu. Nó cũng cho phép xác thực, thực hiện dựa trên tài khoản người dùng [**NTLM**](https://en.wikipedia.org/wiki/NTLM) và [**Active Directory**](https://en.wikipedia.org/wiki/Active_Directory) . Tuy nhiên, việc sử dụng các Plugin mã hóa như vậy khiến nó không tương thích với các chương trình VNC khác. RealVNC đã cung cấp mã hóa **AES** với độ bền cao để thực hiện với vấn đề thương mại, đồng thời nó cũng tích hợp Active Directory. [**Workspot** đã](https://en.wikipedia.org/wiki/Workspot) phát hành các bản vá [mã hóa AES](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) cho VNC. Đối với phần mềm TightVNC, TightVNC không an toàn vì dữ liệu hình ảnh được truyền mà không được mã hóa.

VNC có thể thiết lập đường hầm qua kết nối [**SSH**](https://en.wikipedia.org/wiki/Secure_Shell) hoặc [**VPN**](https://en.wikipedia.org/wiki/Virtual_private_network) để thêm một lớp bảo mật bổ sung với mã hóa mạnh hơn.Đường hầm SSH có thể được tạo từ máy khách [UNIX](https://en.wikipedia.org/wiki/UNIX) , máy khách Microsoft Windows, [máy](https://en.wikipedia.org/wiki/Macintosh) khách [Macintosh](https://en.wikipedia.org/wiki/Macintosh) (bao gồm Mac OS X và [Hệ thống 7](https://en.wikipedia.org/wiki/System_7) trở lên) - và nhiều hệ điều hành khác. Ngoài ra còn có các ứng dụng [phần mềm miễn phí](https://en.wikipedia.org/wiki/Freeware) tạo đường hầm VPN tức thì giữa các máy tính với nhau .

Mối quan tâm về vấn đề bảo mật lớn nhất đã được bổ sung trong việc sử dụng VNC là kiểm tra xem phiên bản được sử dụng, sẽ có yêu cầu cho phép từ chủ sở hữu máy tính từ xa, trước khi ai đó kiểm soát thiết bị của họ hay không. Điều này sẽ tránh trường hợp truy cập trái phép đến máy của họ.

1. **Giao thức RFP :**
   1. **Giới thiệu :**

RFB ("Bộ đệm khung từ xa") là một giao thức đơn giản để truy cập giao diện đồ họa người dùng ở từ xa. Nó có thể áp dụng cho tất cả các hệ thống và ứng dụng cửa sổ, bao gồm cả X11, Windows và Macintosh. RFB là giao thức được sử dụng trong VNC (Virtual Network Computing).

Điểm cuối ở từ xa là nơi người dùng ngồi (tức là màn hình cùng với bàn phím hoặc chuột) được gọi là máy khách hoặc trình xem RFB.

Điểm cuối bắt nguồn từ những thay đổi đối với bộ đệm khung (tức là hệ thống cửa sổ và các ứng dụng) được gọi là máy chủ RFB.

RFB thực sự là một giao thức "Client Thin". Điểm nhấn trong thiết kế của giao thức RFB là ít yêu cầu gì nhiều từ cấu hình máy khách. Bằng cách này, máy khách có thể chạy hết hầu như trong phạm vi phần cứng và nhiệm vụ triển khai máy khách được thực hiện đơn giản nhất có thể.

Nếu một máy khách ngắt kết nối khỏi một máy chủ nhất định và sau đó kết nối lại với cùng một máy chủ đó, trạng thái giao diện của người dùng vẫn được giữ nguyên. Hơn nữa, một điểm cuối máy khách khác nhau có thể được sử dụng để kết nối với cùng một máy chủ RFB. Tại điểm cuối mới, người dùng sẽ thấy chính xác giao diện đồ họa người dùng giống như ở điểm cuối ban đầu.

Trên thực tế, giao diện của ứng dụng người dùng trở nên hoàn toàn di động. Bất cứ nơi nào có kết nối mạng phù hợp, người dùng có thể truy cập vào ứng dụng và trạng thái của các ứng dụng này vẫn được giữ nguyên giữa các lần truy cập từ các vị trí khác nhau. Điều này cung cấp cho người dùng một cái nhìn thống nhất và quen thuộc về cơ sở hạ tầng máy tính ở bất cứ nơi nào họ truy cập.

* 1. **Giao thức hiển thị :**

Màn hình hiển thị của giao thức dựa trên chương trình đồ họa nguyên thủy: "đặt một hình chữ nhật có chứa dữ liệu pixel tại một vị trí x, y cho trước". Thoạt nhìn, đây có vẻ là một cách làm không hiệu quả để vẽ các thành phần giao diện người dùng. Tuy nhiên, việc này cho phép việc mã hóa cho các dữ liệu pixel mang lại cho chúng ta mức độ linh hoạt lớn trong cách đánh đổi các thông số khác nhau như băng thông mạng, tốc độ vẽ máy khách và tốc độ xử lý máy chủ.

Một chuỗi các hình chữ nhật này sẽ được cập nhật trong bộ đệm khung . Bản cập nhật thể hiện sự thay đổi từ trạng thái bộ đệm khung hợp lệ này sang trạng thái bộ đệm khung hợp lệ khác, vì vậy theo một số cách tương tự như một khung video. Các hình chữ nhật trong bản cập nhật thường rời rạc .

Quá trình cập nhật do yêu cầu từ máy khách. Tức là một bản cập nhật chỉ được gửi từ máy chủ đến máy khách để phản hồi lại một yêu cầu rõ ràng từ máy khách. Điều này mang lại cho giao thức một chất lượng thích ứng với các yêu cầu rõ ràng từ máy khách. Máy khách có cấu hình và kết nối càng chậm thì tỷ lệ cập nhật càng chậm.

* 1. **Mô hình màn hình :**

Ở dạng đơn giản nhất, giao thức RFB sử dụng một bộ đệm khung hình chữ nhật duy nhất. Tất c ả các bản cập nhật được chứa trong vùng đệm này và sẽ không được mở rộng ra bên ngoài nó. Một ứng dụng máy khách có chức năng cơ bản là chỉ cần trình bày vùng đệm này cho người dùng, vùng đệm này sẽ xử lí vừa với màn hình của người dùng.

Máy khách và máy chủ RFB nâng cao hơn sẽ có khả năng mở rộng mô hình này và thêm nhiều màn hình hơn nữa. Mục đích là để tạo ra một bản trình bày hoàn chỉnh ở phía máy chủ về bố cục vật lý để tương thích với máy khách. Các ứng dụng có thể sử dụng thông tin này để xác định chính xác đối với viền màn hình của máy khách .

Trong mô hình nhiều màn hình, vẫn chỉ có một bộ đệm khung hình chữ nhật duy nhất và các bản cập nhật bộ đệm khung không bị ảnh hưởng bởi bố cục màn hình. Điều này đảm bảo khả năng tương thích giữa các máy khách cơ bản và máy chủ nâng cao. Màn hình được thêm vào mô hình này và hoạt động giống như khung chữ nhật trong bộ đệm khung.

Một ứng dụng máy khách cơ bản hoạt động như thể có một màn hình duy nhất bao phủ toàn bộ bộ đệm khung. Máy chủ có thể hỗ trợ tối đa 255 màn hình, phải được chứa đầy đủ trong bộ đệm khung hiện tại. Nhiều màn hình có thể chồng lên nhau một phần hoặc hoàn toàn.

Khách hàng có thể yêu cầu thay đổi kích thước bộ đệm khung và bố cục màn hình. Máy chủ có thể tự do phê duyệt hoặc từ chối các yêu cầu này theo ý muốn, nhưng phải luôn thông báo cho máy khách biết kết quả.

Nếu kích thước bộ đệm khung thay đổi vì bất kỳ lý do gì, thì tất cả dữ liệu trong đó sẽ bị vô hiệu và được coi là không xác định.

Chỉ thay đổi bố cục màn hình không ảnh hưởng đến nội dung bộ đệm khung. Do đó, máy khách phải theo dõi các kích thước bộ đệm khung hiện tại và so sánh nó với kích thước nhận được trong hình chữ nhật ExtendedDesktopSize . Chỉ khi chúng khác nhau, nó mới có thể loại bỏ nội dung của bộ đệm khung.

* 1. **Giao thức đầu vào :**

Phía đầu vào của giao thức dựa trên mô hình máy trạm tiêu chuẩn của bàn phím và chuột. Các sự kiện đầu vào chỉ được máy khách gửi đến máy chủ bất cứ khi nào người dùng nhấn một phím hoặc nút chuột hoặc bất cứ khi nào thiết bị chuột được di chuyển. Các sự kiện đầu vào này cũng có thể nhận một số thiết bị I / O không chuẩn khác. Ví dụ: Công cụ nhận dạng chữ viết bằng tay dựa trên bút có thể tạo ra các sự kiện bàn phím.

Nếu bạn có nguồn đầu vào không phù hợp với mô hình máy trạm tiêu chuẩn này, phần mở rộng giao thức General Input Interface (gii) cung cấp khả năng cho các nguồn đầu vào hỗ trợ nhiều thiết bị hơn, tương thích với nhiều nút hơn .

* 1. **Trình bày dữ liệu pixel :**

Tương tác ban đầu giữa máy khách và máy chủ RFB liên quan đến quá trình thương lượng về định dạng và mã hóa mà dữ liệu pixel sẽ được gửi. Thương lượng này được thiết kế để làm cho công việc của người dùng trở nên dễ dàng nhất có thể. Điểm mấu chốt là máy chủ phải luôn có khả năng cung cấp dữ liệu pixel ở dạng mà máy khách mong muốn. Tuy nhiên, nếu máy khách có thể đáp ứng nhiều loại mã hóa khác nhau, nó có thể chọn loại mã hóa có định dạng phù hợp nhất với máy chủ để thực hiện công việc tốt hơn .

Định dạng pixel đề cập đến việc biểu diễn các loại màu theo giá trị pixel. Các định dạng màu phổ biến nhất là "màu thực" 24 bit hoặc 16 bit, trong đó các trường bit có giá trị pixel dịch trực tiếp sang cường độ màu đỏ, xanh lục và xanh lam và "bản đồ màu" 8 bit, nơi có thể ánh xạ tùy ý được sử dụng để chuyển từ các giá trị pixel sang cường độ RGB.

Mã hóa này đề cập đến cách trình bày một hình chữ nhật bằng dữ liệu pixel sẽ được gửi qua máy khách. Mỗi hình chữ nhật của dữ liệu pixel được bắt đầu bằng một tiêu đề cho biết vị trí X, Y của hình chữ nhật trên màn hình, chiều rộng và chiều cao của hình chữ nhật và kiểu mã hóa chỉ định mã hóa của dữ liệu pixel.

* 1. **Phần giao thức mở rộng :**

Có một số cách mà giao thức có thể được mở rộng:

**Các bảng mã hóa mới :**

* Một kiểu mã hóa mới có thể được thêm vào giao thức tương đối dễ dàng trong khi vẫn duy trì khả năng tương thích với các máy khách và máy chủ hiện có.

**Mã hóa giả :**

* Ngoài các mã hóa đặc trưng, máy khách có thể yêu cầu "mã hóa giả" để khai báo với máy chủ rằng nó có thể hỗ trợ một phần mở rộng nhất định trong giao thức. Máy chủ không hỗ trợ mở rộng sẽ đơn giản bỏ qua mã hóa giả.

**Các loại bảo mật mới :**

* Việc thêm một số loại bảo mật mới mang lại sự linh hoạt tối ưu trong việc sửa đổi hành vi của giao thức mà không làm mất khả năng tương thích với các máy khách và máy chủ hiện có.
  1. **Chuỗi mã hóa :**

Mã hóa sử dụng các chuỗi trong các giao thức trước đây, thường là không xác định hoặc đã thay đổi giữa các phiên bản của giao thức. Kết quả là khi triển khai giao thức sẽ sử dụng các bảng mã khác nhau sẽ gây ra vấn đề không tương thích với giao thức. Thông thường những bảng mã đó là ISO 8859-1 hoặc các trang mã Windows.

Nên khuyến khích sử dụng bảng mã UTF8 . Điều này cho phép hỗ trợ Unicode đầy đủ nhưng vẫn giữ được khả năng tương thích tốt với các giao thức RFB cũ hơn.

Tất cả các máy khách và máy chủ phải luôn sẵn sàng để nhận các chuỗi UTF-8 không hợp lệ. Những điều này có thể xảy ra do lỗi. Cả hai trường hợp đều không dẫn đến mất đồng bộ hóa giao thức.

* 1. **Thông điệp từ giao thức :**

Giao thức **RFB** có thể hoạt động trên bất kỳ nền tảng nào. Thông thường, nó được sử dụng qua kết nối **TCP / IP**. Có ba giai đoạn đối với giao thức. Đầu tiên là giai đoạn bắt tay, mục đích của nó là đồng ý về phiên bản giao thức và loại bảo mật sẽ được sử dụng. Giai đoạn thứ hai là giai đoạn khởi tạo, là nơi mà máy khách và máy chủ sẽ trao đổi các thông điệp **ClientInit**và **ServerInit**. Giai đoạn cuối cùng là tương tác giao thức bình thường. Máy khách có thể gửi bất kỳ thông điệp nào nó muốn và kết quả là có thể nhận được thông báo từ máy chủ.

Giao thức sẽ mô tả các thông điệp bằng cách sử dụng các loại dữ liệu cơ bản như : U8, U16, U32 đại diện cho kiểu số nguyên không dấu có 8, 16, 32 bit và S8, S16, S32 đại diện cho kiểu số nguyên có dấu có 8, 16, 32 bit .

* + 1. **Thông điệp bắt tay :**

1. **Protocol Version :**

Quá trình bắt tay bắt đầu bằng cách máy chủ gửi cho máy khách một thông điệp ProtocolVersion. Điều này cho phép máy khách biết được phiên bản giao thức RFB cao nhất, mà máy chủ có thể hỗ trợ. Sau đó, máy khách sẽ trả lời bằng một thông báo tương tự và cho biết số phiên bản của giao thức mà có nó sử dụng. Máy khách không bao giờ được yêu cầu phiên bản giao thức cao hơn phiên bản do máy chủ cung cấp.

Việc bổ sung kiểu mã hóa mới hoặc kiểu mã hóa giả không yêu cầu thay đổi phiên bản giao thức, vì máy chủ có thể đơn giản bỏ qua các mã hóa mà nó không hiểu.

Thông báo ProtocolVersion bao gồm 12 byte được hiểu là một chuỗi ký tự ASCII ở định dạng "RFB xxx.yyy \ n" trong đó xxx và yyy là số phiên bản chính và phụ .

|  |  |
| --- | --- |
| No of Byte | Value |
| 12 | "RFB 003.003\n" (hex 52 46 42 20 30 30 33 2e 30 30 33 0a) |

**Thông báo Protocol Version RFB 3.3**

|  |  |
| --- | --- |
| No of Byte | Value |
| 12 | "RFB 003.007\n" (hex 52 46 42 20 30 30 33 2e 30 30 37 0a) |

**Thông báo Protocol Version RFB 3.7**

|  |  |
| --- | --- |
| No of Byte | Value |
| 12 | "RFB 003.008\n" (hex 52 46 42 20 30 30 33 2e 30 30 38 0a) |

**Thông báo Protocol Version RFB 3.8**

1. **Bảo mật :**

Khi phiên bản giao thức đã được thương lượng xong thì sau đó máy chủ và máy khách sẽ phải tiếp tục thương lượng về loại bảo mật .

Giao thức RFB từ phiên bản 3.7 trở đi :

Máy chủ sẽ liệt kê các loại bảo mật mà nó hỗ trợ gửi đến máy khách :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 1 | U8 | Number-of-security-types |
| Number-of-security-types | U8 array | Security-types |

**Thông điệp từ máy chủ.**

Nếu máy chủ liệt kê ít nhất một kiểu bảo mật hợp lệ mà được máy khách hỗ trợ, máy khách sẽ gửi lại một byte duy nhất cho biết kiểu bảo mật này sẽ được sử dụng trong kết nối giữa máy khách và máy chủ:

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 1 | U8 | Security-type |

**Thông điệp từ máy khách gửi đến máy chủ.**

Nếu **Number-of-security-type** bằng 0 do một vài lý do nào đó, thì kết nối sẽ không thành công (ví dụ: máy chủ không thể hỗ trợ phiên bản giao thức mong muốn). Tiếp theo là một chuỗi mô tả lý do (trong đó một chuỗi được chỉ định là độ dài theo sau là nhiều ký tự ASCII):

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 4 | U32 | Reason-length |
| Reason-length | U8 array | Reason-string |

**Máy chủ sẽ đóng kết nối và sau đó sẽ gửi thông điệp bằng kiểu String.**

Các loại bảo mật đã được đăng ký khác là:

|  |  |
| --- | --- |
| Number | Name |
| 3-4 | RealVNC |
| 5 | RA2 |
| 6 | RA2ne |
| 7-15 | RealVNC |
| 17 | Ultra |
| 18 | TLS |
| 20 | SASL |
| 21 | MDH5 hash authentication |
| 22 | xvp |
| 23 | Secure Tunel |
| 24 | Integared SSH |
| 30-35 | Apple Inc. |

1. **Kết quả bảo mật :**

Sau khi đã thực hiện các bước trên thì máy chủ sẽ gửi một từ để thông báo cho máy khách biết bắt tay bảo mật có thành công hay không.

|  |  |  |  |
| --- | --- | --- | --- |
| No of bytes | Type | Value | Description |
| 4 | U32 |  | Status |
|  |  | 0 | Ok |
|  |  | 1 | Failed |
|  |  | 2 | Failed, too many attempts |

**Thông điệp từ máy chủ gửi đến máy khách .**

Nếu quá trình trên thành công , thì giao thức chuyển sang giai đoạn khởi tạo .

Và nếu không thành công , thì máy chủ sẽ gửi một chuỗi vì lí do thất bại , và sau đó đóng kết nối.

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 4 | U32 | Reason-lenght |
| Reason-length | U8 array | Reason-string |

**Thông điệp lí do thất bại được gửi từ máy chủ sang máy khách .**

* + 1. **Các loại bảo mật :**

1. **Không có :**

Khi không có loại bảo mật cần xác thực thì dữ liệu giao thức sẽ được gửi mà không được mã hóa.

Phiên bản 3.8 trở đi giao thức sẽ có thông báo SecurityResult .

Phiên bản 3.3 và 3.7 không có SecurityResult mà chuyển sang giai đoạn khởi tạo.

1. **Xác thực VNC :**

Xác thực VNC khi sử dụng thì dữ liệu giao thức sẽ được gửi đi đến máy khách. Máy chủ sẽ gửi một 16 byte Challenge ngẫu nhiên theo bảng dưới đây :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 16 | U8 | Challenge |

Máy khách sẽ mã hóa Challenge bằng DES, sử dụng mật khẩu do người dùng cung cấp làm khóa. Mật khẩu dài hơn 64 bit, theo yêu cầu của DES chỉ đơn giản là bị cắt bớt. Nếu mật khẩu ngắn hơn yêu cầu .thì khóa sẽ được them vào bằng các số 0.

Mỗi 8 byte của Challenge được mã hóa độc lập (tức là chế độ ECB) và được gửi trở lại máy chủ:

|  |  |  |
| --- | --- | --- |
| No of Bytes | Type | Description |
| 16 | U8 | Challenge |

1. **Vấn đề bảo mật chặt chẽ :**

Bảo mật chặt chẽ là một phần mở rộng của giao thức, chúng cho phép thực hiện ba điều:

* Xây dựng đường hầm thông qua kết nối như SSH hay VPN.
* Xác thực : Bảo mật chặt chẽ cho phép xác thực linh hoạt giữa máy khách, và máy chủ .
* Khả năng của máy chủ : Bước cuối cùng, bảo mật chặt chẽ sẽ mở rộng và gửi thông báo ServerInit , nó cho phép máy chủ thông báo cho máy khách các khả năng của máy chủ về mã hóa và các loại thông báo được hỗ trợ.

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 4 | U32 | Number-of-tunnels |

Sau khi loại bảo mật chặt chẽ đã được chọn, máy chủ sẽ bắt đầu gửi danh sách các đường hầm được hỗ trợ, theo thứ tự ưu tiên :

**Danh sách các đường hầm máy chủ gửi đi .**

Trong đó **Number-of-tunnels** có dạng như sau :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 16 | CAPABILITY | tunnel |

Và **CAPABILITY** có dạng như sau :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 4 | S32 | code |
| 4 | U8 array | Vendor |
| 8 | U8 array | Signature |

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Vendor | Signature | Description |
| 0 | “TIGH” | "NOTUNNEL" | No tunneling |

Sau đó Tunel Capabilities sẽ được đăng kí với máy khách :

Sau đây là 1 số Authentication capabilities được sử dụng phổ biến :

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Vendor | Signature | Description |
| 1 | “STDV” | "NOAUTH\_\_" | None |
| 2 | “STDV” | "VNCAUTH\_" | VNC Authentication |
| 19 | “VENC” | "VENCRYPT" | [VeNCrypt](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#vencrypt) |
| 20 | “GTKV” | "SASL\_\_\_\_" | Simple Authentication and Security Layer (SASL) |
| 129 | “TGHT” | "ULGNAUTH" | [Tight Unix Login Authentication](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#tight-unix-login-authentication) |
| 130 | “TGHT” | "XTRNAUTH" | External Authentication |

Sau khi thực hiện quá trình đăng kí đường hầm và xác thực như trên thì bảo mật chặt chẽ sẽ được mở rộng và sẽ gửi ServerInit từ máy chủ đến máy khách.Vấn đề bảo mật chặt chẽ tăng thêm độ bảo mật tránh các trường không may gây ra .

1. **VeNCrypt :**

Kiểu bảo mật VeNCrypt là một phương pháp xác thực chung bao gồm nhiều kiểu xác thực con .

Sau loại bảo mật VeNCrypt được chọn ở máy khách, thì máy chủ sẽ gửi phiên bản VeNCrypt cao nhất mà nó có thể hỗ trợ. Mặc dù tồn tại hai phiên bản 0,1 và 0,2 nhưng chúng tôi sẽ chọn phiên bản mới nhất là 0.2 để trình bày :

|  |  |  |  |
| --- | --- | --- | --- |
| No of bytes | Type | Value | Description |
| 1 | U8 | 0 | Major version number |
| 1 | U8 | 2 | Minor version number |

**Bảng này mô tả phiên bản 0.2 của VeNCrypt từ máy chủ đến máy khách.**

Sau đó, máy khách gửi lại phiên bản VeNCrypt cao nhất mà nó có thể hỗ trợ :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 1 | U8 | Major version number |
| 1 | U8 | Minor version number |

**Bảng này mô tả thông điệp phản hồi từ máy khách đến máy chủ .**

Sau đó máy chủ sẽ phản hồi một byte cho máy khách để biết mọi thứ có ổn không. Giá trị khác 0 có nghĩa là lỗi và kết nối sẽ bị đóng. Giá trị bằng không có nghĩa là thành công.

* + 1. **Thông báo khởi tạo :**

Khi máy khách và máy chủ đã thỏa thuận được với nhau ,giao thức sẽ chuyển sang giai đoạn khởi tạo. Máy khách gửi một thông điệp ClientInit đến máy chủ, sau đó máy chủ gửi một thông điệp ServerInit đến máy khách .

1. **ClientInit :**

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 1 | U8 | shared-flag |

**Bảng trên mô tả thông điệp Clientit.**

Shared-flag có giá trị khác 0 (true) nếu máy chủ cố gắng chia sẻ máy tính bằng cách kết nối các máy khách khác, zero (false) nếu máy chủ cấp quyền truy cập độc quyền cho máy khách này bằng cách ngắt kết nối tất cả các máy khách khác.

1. **ServerInit :**

Sau khi nhận được thông báo ClientInit , máy chủ sẽ gửi một thông báo ServerInit. Điều này cho máy khách biết được chiều rộng và chiều cao của bộ đệm khung ở máy chủ, định dạng pixel và tên màn hình được liên kết :

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 2 | U16 | Framebuffer-width |
| 2 | U16 | Framebuffer-height |
| 16 | PIXEL\_FORMAT | Server-pixel-format |
| 4 | U32 | Name-lenght |
| Name-lenght | U8 array | Name-string |

**Bảng trên mô tả thông điệp ServerInit .**

PIXEL\_FORMAT được định nghĩa là:

|  |  |  |
| --- | --- | --- |
| No of bytes | Type | Description |
| 1 | U8 | Bit-per-pixel |
| 1 | U8 | Depth |
| 1 | U8 | Big-endian-flag |
| 1 | U8 | True-colour-flag |
| 2 | U16 | red-max |
| 2 | U16 | green-max |
| 2 | U16 | Blue-max |
| 1 | U8 | Red-shift |
| 1 | U8 | Green-shift |
| 1 | U8 | Blue-shift |
| 3 |  | Padding |

* + 1. **Thông báo từ máy khách đến máy chủ :**

Loại thông báo từ máy khách đến máy chủ mà tất cả máy chủ phải hỗ trợ là:

| **Number** | **Name** |
| --- | --- |
| 0 | [SetPixelFormat](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#setpixelformat) |
| 2 | [SetEncodings](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#setencodings) |
| 3 | [FramebufferUpdateRequest](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#framebufferupdaterequest) |
| 4 | [KeyEvent](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#keyevent) |
| 5 | [PointerEvent](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#pointerevent) |
| 6 | [ClientCutText](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#clientcuttext) |

Và loại thông báo tùy chọn là :

| **Number** | **Name** |
| --- | --- |
| 7 | FileTransfer |
| 8 | SetScale |
| 9 | SetServerInput |
| 10 | SetSW |
| 11 | TextChat |
| 12 | KeyFrameRequest |
| 13 | KeepAlive |
| 14 | Possibly used in UltraVNC |
| 15 | SetScaleFactor |
| 16-19 | Possibly used in UltraVNC |
| 20 | RequestSession |
| 21 | SetSession |
| 80 | NotifyPluginStreaming |
| 127 | VMware |
| 128 | Car Connectivity |
| 150 | [EnableContinuousUpdates](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#enablecontinuousupdates) |
| 248 | [ClientFence](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#clientfence) |
| 249 | OLIVE Call Control |
| 250 | [xvp Client Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#xvp-client-message) |
| 251 | [SetDesktopSize](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#setdesktopsize) |
| 252 | tight |
| 253 | [gii Client Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#gii-client-message) |
| 254 | VMware |
| 255 | [QEMU Client Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#qemu-client-message) |

Lưu ý rằng trước khi gửi tin nhắn với loại tin nhắn tùy chọn, máy khách phải xác định rằng máy chủ hỗ trợ tiện ích mở rộng liên quan bằng cách nhận một số xác nhận dành riêng cho tiện ích mở rộng từ máy chủ.

1. **Set PixelFormat :**

Quá trình đặt giá trị định dạng pixel sẽ được máy khách gửi đến máy chủ trong thông báo FramebufferUpdate. Nếu máy khách không gửi thông báo SetPixelFormat thì máy chủ sẽ gửi các giá trị pixel ở định dạng tự nhiên như được chỉ định trong thông báo [ServerInit](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#serverinit) ( [ServerInit](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#serverinit) ).

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 0 | message-type |
| 3 |  |  | padding |
| 16 | PIXEL\_FORMAT |  | pixel-format |

**Thông điệp máy chủ gửi đến máy khách.**

Thông tin PIXEL\_FORMAT tại ServerInit

1. **Set Encodings :**

Đặt kiểu kiểu mã hóa mà máy chủ có thể gửi dữ liệu pixel đến máy khách. Thứ tự của các kiểu mã hóa được đưa ra trong thông báo này là một gợi ý cho máy khách về tùy chọn của nó (các kiểu mã hóa được sắp xếp theo thứ tự ưu tiên). Máy chủ có thể chọn hoặc không sử dụng gợi ý này. Dữ liệu pixel có thể luôn được gửi ở dạng mã hóa thô khi không được chỉ định rõ ràng ở đây.

Ngoài các mã hóa phổ biến, máy khách có thể yêu cầu "mã hóa giả" để khai báo với máy chủ rằng nó hỗ trợ một số phần mở rộng nhất định cho giao thức. Máy chủ không hỗ trợ tiện ích mở rộng sẽ đơn giản bỏ qua mã hóa giả.

1. **Framebuffer Update Request :**

Gửi thông báo cho máy chủ biết rằng máy khách đang yêu cầu Framebuffer xác định bởi x-vị trí , y-vị trí , chiều rộng và chiều cao . Máy chủ thường phản hồi Framebuffer Update Request bằng cách gửi Framebuffer Update . Tuy nhiên, lưu ý rằng một Frame Buffer Update duy nhất có thể được gửi trả lời cho nhiều Framebuffer Update Requests .Ví dụ ( Trong trường hợp nhiều máy khách kết nối đến máy chủ ) .

Máy chủ giả định rằng máy khách sẽ giữ một bản sao tất cả các phần của bộ đệm khung mà nó yêu cầu. Điều này có nghĩa là thông thường máy chủ chỉ cần gửi các bản cập nhật liên tục cho máy khách.

Trong trường hợp máy khách nhanh, máy khách có thể điều chỉnh tốc độ gửi Framebuffer Update Requests*g*ia tăng để tránh làm hỏng mạng.

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 3 | message-type |
| 1 | U8 |  | incremental |
| 2 | U16 |  | x-position |
| 2 | U16 |  | y-position |
| 2 | U16 |  | width |
| 2 | U16 |  | height |

**Thông điệp từ máy chủ sẽ được gửi đến máy khách.**

Yêu cầu cho khu vực nằm một phần bên ngoài bộ đệm khung hiện tại sẽ phải được cắt sao cho nó vừa với kích thước bộ đệm khung. Lưu ý rằng một vùng trống vẫn có thể lấy FramebufferUpdate mặc dù bản cập nhật đó sẽ chỉ chứa các mã giả.

1. **KeyEvent :**.

Có hai sự kiện được quan tâm ở trong mục này là : nhấn và thả .Nếu nhấn giá trị sẽ khác 0 (true) , và được thả giá trị sẽ bằng 0 (false). Bản thân khóa được chỉ định bằng cách sử dụng các giá trị "keyym" do hệ thống cửa sổ X xác định.

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 4 | message-type |
| 1 | U8 |  | down-flag |
| 2 |  |  | padding |
| 4 | U32 |  | key |

**Mô tả sự kiện nhập từ bàn phím của máy khách gửi đến server.**

Tự động lặp lại các phím khi nhấn giữ phím phải được xử lý trên máy khách. Lý do là độ trễ cao trên mạng có thể làm cho nó có vẻ như một Key đang được giữ trong một thời gian rất dài, nhưng vấn đề là thông báo KeyEventphát hành nút đã bị trì hoãn.

Một số khóa phổ biến khác là:

| **Key name** | **Keysym value** |
| --- | --- |
| BackSpace | 0xff08 |
| Tab | 0xff09 |
| Return or Enter | 0xff0d |
| Escape | 0xff1b |
| Insert | 0xff63 |
| Delete | 0xffff |
| Home | 0xff50 |
| End | 0xff57 |
| Page Up | 0xff55 |
| Page Down | 0xff56 |
| Left | 0xff51 |
| Up | 0xff52 |
| Right | 0xff53 |
| Down | 0xff54 |
| F1 | 0xffbe |
| F2 | 0xffbf |
| F3 | 0xffc0 |
| F4 | 0xffc1 |
| ... | ... |
| F12 | 0xffc9 |
| Shift (left) | 0xffe1 |
| Shift (right) | 0xffe2 |
| Control (left) | 0xffe3 |
| Control (right) | 0xffe4 |
| Meta (left) | 0xffe7 |
| Meta (right) | 0xffe8 |
| Alt (left) | 0xffe9 |
| Alt (right) | 0xffea |

1. P**ointer Event :**

Nó cho biết chuyển động của con trỏ hoặc nhấn hoặc thả nút con trỏ. Con trỏ hiện tại là *(*x-position*,*y-position*)* và trạng thái hiện tại của nút là từ 1 đến 8 được đại diện bởi bit 0-7 của button-mask tương ứng, 0 có nghĩa là lên, 1 có nghĩa là xuống .

Trên chuột thông thường, các nút 1, 2 và 3 tương ứng với các nút trái, giữa và phải trên chuột. Trên chuột bánh xe, mỗi bước của bánh xe của chuột được biểu diễn bằng cách nhấn và thả một nút nhất định. Nút 4 có nghĩa là lên, nút 5 có nghĩa là xuống, nút 6 có nghĩa là trái và nút 7 có nghĩa là phải.

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 5 | message-type |
| 1 | U8 |  | button-mask |
| 2 | U16 |  | x-position |
| 2 | U16 |  | y-position |

**Bảng trên mô tả thông điệp từ con trỏ của máy khách gửi đến máy chủ .**

1. **Client Cutext :**

Máy khách có văn bản được định dạng theo ISO 8859-1 (Latin-1) trong bộ đệm cắt của nó. Phần cuối của các dòng được biểu thị bằng ký tự dòng cấp dữ liệu dòng mới (giá trị 10). Không cần ký tự xuống dòng (giá trị 13).

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 6 | message-type |
| 3 |  |  | padding |
| 4 | U32 |  | length |
| length | U8 array |  | text |

* + 1. **Thông báo từ máy chủ gửi đến máy khách :**

Loại thông báo từ máy chủ gửi đến máy khách mà tất cả các máy khách phải hỗ trợ là:

| **Number** | **Name** |
| --- | --- |
| 0 | [FramebufferUpdate](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#framebufferupdate) |
| 1 | [SetColourMapEntries](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#setcolourmapentries) |
| 2 | [Bell](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#bell) |
| 3 | [ServerCutText](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#servercuttext) |

Các loại thông báo tùy chọn là:

| **Number** | **Name** |
| --- | --- |
| 4 | ResizeFrameBuffer |
| 5 | KeyFrameUpdate |
| 6 | Possibly used in UltraVNC |
| 7 | FileTransfer |
| 8-10 | Possibly used in UltraVNC |
| 11 | TextChat |
| 12 | Possibly used in UltraVNC |
| 13 | KeepAlive |
| 14 | Possibly used in UltraVNC |
| 15 | ResizeFrameBuffer |
| 127 | VMware |
| 128 | Car Connectivity |
| 150 | [EndOfContinuousUpdates](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#endofcontinuousupdates) |
| 173 | ServerState |
| 248 | [ServerFence](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#serverfence) |
| 249 | OLIVE Call Control |
| 250 | [xvp Server Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#xvp-server-message) |
| 252 | tight |
| 253 | [gii Server Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#gii-server-message) |
| 254 | VMware |
| 255 | [QEMU Server Message](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#qemu-server-message) |

Lưu ý rằng trước khi gửi tin nhắn với loại tin nhắn tùy chọn, máy chủ phải xác định rằng máy khách sẽ có hỗ trợ tiện ích mở rộng liên quan, bằng cách nhận một số xác nhận dành riêng cho tiện ích mở rộng từ máy khách .Thường là một yêu cầu cho một mã hóa giả nhất định .

1. **Framebuffer Update :**

Bản cập nhật của bộ đệm khung bao gồm một chuỗi dữ liệu pixel hình chữ nhật mà máy khách sẽ đưa vào bộ đệm khung của nó. Nó sẽ được máy chủ gửi để phản hồi lại một FramebufferUpdateRequesttừ máy khách .

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 0 | message-type |
| 1 |  |  | padding |
| 2 | U16 |  | number-of-rectangles |

**Bảng trên mô tả dữ liệu sẽ được máy chủ gửi đến máy khách.**

Dữ liệu của hình chữ nhật là :

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 2 | U16 | x-position |
| 2 | U16 | y-position |
| 2 | U16 | width |
| 2 | U16 | height |
| 4 | S32 | encoding-type |

Lưu ý rằng bản cập nhật bộ đệm khung sẽ đánh dấu sự chuyển đổi từ trạng thái bộ đệm khung hợp lệ này sang trạng thái bộ đệm khung khác. Điều đó có nghĩa là một bản cập nhật duy nhất xử lý tất cả FramebufferUpdateRequestđã nhận được từ máy khách cho đến thời điểm bản cập nhật được gửi đi.

1. **Set Colour Map Entries :**

Khi định dạng pixel sử dụng "color-map", thông báo này cho máy khách biết rằng các giá trị pixel được chỉ định nên được ánh xạ tới các cường độ RGB nhất định.

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 1 | message-type |
| 1 |  |  | padding |
| 2 | U16 |  | first-colour |
| 2 | U16 |  | number-of-colours |

**Thông điệp được máy chủ gửi đến máy khách.**

Sau đó là dữ liệu màu sắc sau đây :

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 2 | U16 | red |
| 2 | U16 | green |
| 2 | U16 | blue |

1. **Bell :**

Chuông kêu nếu có sự kết nối từ máy khách :

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 2 | message-type |

1. **ServerCutText :**

Máy chủ có định dạng văn bản theo ISO 8859-1 (Latin-1) trong bộ đệm cắt của nó. Phần cuối của các dòng được biểu thị bằng ký tự dòng cấp dữ liệu / dòng mới (giá trị 10). Không cần ký tự xuống dòng (giá trị 13).

| **No. of bytes** | **Type** | **[Value]** | **Description** |
| --- | --- | --- | --- |
| 1 | U8 | 3 | message-type |
| 3 |  |  | padding |
| 4 | U32 |  | length |
| length | U8 array |  | text |

* + 1. **Mã hóa :**

Các mã hóa được xác định trong bản báo cáo này là:

| **Number** | **Name** |
| --- | --- |
| 0 | [Raw Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#raw-encoding) |
| 1 | [CopyRect Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#copyrect-encoding) |
| 2 | [RRE Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#rre-encoding) |
| 4 | [CoRRE Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#corre-encoding) |
| 5 | [Hextile Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#hextile-encoding) |
| 6 | [zlib Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#zlib-encoding) |
| 7 | [Tight Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#tight-encoding) |
| 8 | [zlibhex Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#zlibhex-encoding) |
| 16 | [ZRLE Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#zrle-encoding) |
| -260 | [Tight PNG Encoding](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#tight-png-encoding) |

Các loại mã hóa đã đăng ký khác là:

| **Number** | **Name** |
| --- | --- |
| 9 | Ultra |
| 10 | Ultra2 |
| 15 | TRLE |
| 17 | Hitachi ZYWRLE |
| 20 | H.264 |
| 21 | JPEG |
| 22 | JRLE |
| 1000 to 1002 | Apple Inc. |
| 1011 | Apple Inc. |
| 1024 to 1099 | RealVNC |
| 1100 to 1105 | Apple Inc. |
| -1 to -22 | Tight options |
| -33 to -218 | Tight options |
| -219 to -222 | Historical libVNCServer use |
| -225 | PointerPos |
| -226 to -238 | Tight options |
| -241 to -246 | Tight options |
| -262 to -272 | QEMU |
| -273 to -304 | VMware |
| -306 | popa |
| -310 | OLIVE Call Control |
| -311 | ClientRedirect |
| -523 to -528 | Car Connectivity |
| 0x48323634 | VA H.264 |
| 0x574d5600 to 0x574d56ff | VMware |
| 0xc0a1e5cf | PluginStreaming |
| 0xfffe0000 | KeyboardLedState |
| 0xfffe0001 | SupportedMessages |
| 0xfffe0002 | SupportedEncodings |
| 0xfffe0003 | ServerIdentity |
| 0xfffe0004 to 0xfffe00ff | libVNSServer |
| 0xffff0000 | Cache |
| 0xffff0001 | CacheEnable |
| 0xffff0002 | XOR zlib |
| 0xffff0003 | XORMonoRect zlib |
| 0xffff0004 | XORMultiColor zlib |
| 0xffff0005 | SolidColor |
| 0xffff0006 | XOREnable |
| 0xffff0007 | CacheZip |
| 0xffff0008 | SolMonoZip |
| 0xffff0009 | UltraZip |
| 0xffff8000 | ServerState |
| 0xffff8001 | EnableKeepAlive |
| 0xffff8002 | FTProtocolVersion |
| 0xffff8003 | Session |

1. **Raw Encoding :**

Raw Encoding là kiểu dữ liệu mã hóa đơn giản nhất. Trong trường hợp này, dữ liệu bao gồm các giá trị pixel của (**chiều rộng \* chiều cao**) trong đó chiều rộng và chiều cao là chiều rộng và chiều cao của hình chữ nhật . Các giá trị chỉ đại diện cho mỗi pixel theo thứ tự dòng được quét từ trái sang phải. Tất cả các máy khách RFB sẽ chịu trách nhiệm xử lý mã hóa dữ liệu pixel này và các máy chủ RFB chỉ nên tạo mã hóa thô khi máy khách yêu cầu cụ thể một số kiểu mã hóa này.

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| width \* height \* bytesPerPixel | PIXEL array | pixels |

**Bảng trên thể hiện đoạn dữ liệu đã được mã hóa từ máy chủ gửi đến máy khách.**

1. **CoppyRect Encoding :**

Mã hóa theo kiểu CoppyRect (sao chép hình chữ nhật) là một loại mã hóa rất đơn giản và hiệu quả có thể được sử dụng khi máy khách đã có cùng một dữ liệu pixel ở nơi khác trong bộ đệm khung của nó.

Mã hóa này chỉ đơn giản bao gồm một tọa độ X, Y. Điều này cung cấp một vị trí trong bộ đệm khung mà từ đó máy khách có thể sao chép hình chữ nhật của dữ liệu pixel

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 2 | U16 | src-x-position |
| 2 | U16 | src-y-position |

**Thể hiện dữ liệu được mã hóa theo kiểu CoppyRect được gửi đến máy khách.**

1. **RRE Encoding :**

RRE là viết tắt của từ Rise-And-Run-Length Encoding và như tên gọi của nó, về cơ bản nó là một dạng mã hóa theo 2 chiều độ dài của chuỗi. Các hình chữ nhật được mã hóa theo RRE gửi đến máy khách có thể được hiển thị ngay lập tức và hiệu quả nhất. RRE không thích hợp cho các máy tính để bàn phức tạp, nhưng có thể hữu ích trong một số trường hợp.

Ý tưởng cơ của RRE là phân vùng dữ liệu pixel của hình chữ nhật thành các tiểu vùng hình chữ nhật (hình chữ nhật con), mỗi tiểu vùng bao gồm có một giá trị Pixel duy nhất và liên hợp với nhau tạo thành một hình chữ nhật ban đầu.

Mã hóa bao gồm giá trị pixel nền, Vb (thường là giá trị pixel phổ biến nhất trong hình chữ nhật) và số đếm N , theo sau là danh sách N  hình chữ nhật con, mỗi hình chữ nhật này bao gồm một tuple < v , x , y , w , h > v (! = Vb ) là giá trị pixel, ( x , y ) là tọa độ của subrectangle tương đối so với góc trên bên trái của hình chữ nhật, và ( w , h) là chiều rộng và chiều cao của hình chữ nhật con. Máy khách có thể kết xuất hình chữ nhật ban đầu bằng cách vẽ một hình chữ nhật được tô đầy giá trị pixel nền và sau đó vẽ một hình chữ nhật được tô tương ứng với mỗi hình chữ nhật con.

Dữ liệu bắt đầu bằng tiêu đề:

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 4 | U32 | number-of-subrectangles |
| bytesPerPixel | PIXEL | background-pixel-value |

**Bảng này thể hiện dữ liệu mã hóa bằng RRE.**

Ví dụ về một số cấu trúc sau đây :

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| bytesPerPixel | PIXEL | subrect-pixel-value |
| 2 | U16 | x-position |
| 2 | U16 | y-position |
| 2 | U16 | width |
| 2 | U16 | height |

1. **CoRRE Encoding :**

CoRRE là viết tắt của từ Rise-And-Run-Length Encoding và như tên gọi của nó, nó là một biến thể của [mã hóa RRE](https://github.com/rfbproto/rfbproto/blob/master/rfbproto.rst#rre-encoding) ở trên.

Sự khác biệt duy nhất giữa CoRRE và RRE là vị trí, chiều rộng và chiều cao của các hình tam giác con được giới hạn ở mức tối đa 255 pixel. Do đó, máy chủ cần tạo ra một số hình chữ nhật để bao phủ một khu vực lớn hơn.

Dữ liệu bắt đầu bằng tiêu đề:

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 4 | U32 | number-of-subrectangles |
| bytesPerPixel | PIXEL | background-pixel-value |

**Bảng trên thể hiện dữ liệu được mã hóa bằng CoRRE.**

Đây là những ví dụ về cấu trúc :

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| bytesPerPixel | PIXEL | subrect-pixel-value |
| 1 | U8 | x-position |
| 1 | U8 | y-position |
| 1 | U8 | width |
| 1 | U8 | height |

**6.8.7 Mã hóa giả :**

* 1. **Mã hóa giả chất lượng JPEG :**

Khi máy khách chỉ định chất lượng hình ảnh mong muốn từ bộ mã hóa JPEG. Số mã hóa -23 bằng mã hóa JPEG ở chất lượng cao và bằng -32 mã hóa chất JPEG ở chất lượng thấp. Chất lượng thấp có thể hữu ích trong các tình huống chất lượng kết nối mạng kém, băng thông thấp. Nếu mức chất lượng JPEG không được chỉ định, thì **JpegCompression** sẽ không được sử dụng.

Mức chất lượng ảnh liên quan đến việc nén dữ liệu bị mất nên do đó cần phải cài đặt cân bằng giữa chất lượng hình ảnh và băng thông.Đặc điểm của loại mã hóa này là tính thiếu ổn định vì nó phải phụ thuộc vào đường mạng và băng thông cho phép.

* 1. **Mã hóa giả con trỏ :**

Khi máy khác yêu cầu mã hóa giả **Con trỏ** . Máy chủ sẽ đặt hình dạng con trỏ bằng cách gửi một hình chữ nhật giả với mã hóa giả Con trỏ như một phần của bản cập nhật trong FramebufferUpdate. Các hình chữ nhật có x-vị trí và y-vị trí nó sẽ chỉ ra hotspot của con trỏ, chiều rộng và chiều cao sẽ chỉ ra chiều rộng và chiều cao của con trỏ theo dữ liệu pixel.

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| width \* height \* bytesPerPixel | PIXEL array | cursor-pixels |
| floor((width + 7) / 8) \* height | U8 array | bitmask |

**Bảng này thể hiện dữ liệu mã hóa của con trỏ.**

* 1. **Mã hóa giả con trỏ X :**

Khi máy khách yêu cầu mã hóa giả con trỏ X Máy chủ đặt hình dạng con trỏ bằng cách gửi một hình chữ nhật giả với mã hóa giả con trỏ X như một phần của bản cập nhật Framebuffer Update.

Các hình chữ nhật có x-vị trí và y-vị trí chỉ ra hotspot của con trỏ, và chiều rộng*và*chiều caochỉ ra chiều rộng và chiều cao của con trỏ theo pixel.

| **No. of bytes** | **Type** | **Description** |
| --- | --- | --- |
| 1 | U8 | primary-r |
| 1 | U8 | primary-g |
| 1 | U8 | primary-b |
| 1 | U8 | secondary-r |
| 1 | U8 | secondary-g |
| 1 | U8 | secondary-b |
| floor((width + 7) / 8) \* height | U8 array | bitmap |
| floor((width + 7) / 8) \* height | U8 array | bitmask |

**Bảng này thể hiện dữ liệu mã hóa của con trỏ X.**

* 1. **Mã hóa giả kích thước DesktopSize**

Khi máy khách yêu cầu mã hóa giả **DesktopSize**thì nó sẽ đảm bảo khả năng thích ứng với sự thay đổi về chiều rộng và chiều cao của bộ đệm khung.

Máy chủ khi thay đổi kích thước màn hình bằng cách nó sẽ gửi một hình chữ nhật giả với mã hóa giả DesktopSize. Các hình chữ nhật giả có x-vị trí và y vị trí sẽ được bỏ qua, và chiều rộng và chiều cao chỉ ra chiều rộng mới và chiều cao mớicủa Framebuffer. Không có thêm dữ liệu nào được liên kết với hình chữ nhật giả.

**Chương 2 : Tổng Quan Về REAL.VNC**

1. **RealVNC là gì ?**

RealVNC là một công ty cung cấp phần mềm [truy cập từ xa](https://en.wikipedia.org/wiki/Remote_desktop_software) giữa các máy tính với nhau .Phần mềm bao gồm (VNC Server) được cài đặt ở máy chủ và (VNC Viewer) được cài đặt ở máy khách. Chúng được thực dựa trên giao thức mạng máy tính ảo (VNC) để thao tác, điều khiển màn hình của máy tính khác ở từ xa. Phần mềm này cho phép bạn kết nối với một máy tính từ khắp mọi nơi trên giới.



**Logo RealVNC.**

* **Tính đơn giản của VNC :**
* Dễ dàng triển khai và quản lý .
* Đáp ứng một cách trực quan cho người dùng .
* Người dùng có thể sử dụng một cách dễ dàng mà không cần được đào tạo bài bản .
* Đáp ứng mọi tính năng mà người dùng mong muốn .
* **Tính linh hoạt trong VNC :**
* Truy cập dưới sự giám sát và không giám sát của máy chủ .
* Dễ dàng truy cập, kết nối đến máy chủ .
* Có thể sử dụng để truyền tệp, in dữ liệu, và trò chuyện mới nhau giữa máy khách và máy chủ .
* **Đảm bảo an toàn bảo mật cao :**
* Sử dụng phiên bản mã hóa AES 256-bit .
* Xác thực đa yếu tố .
* Phân quyền cho các máy khách một cách đa dạng .

RealVNC không chỉ là một công cụ CNTT, VNC connect còn dành cho doanh nghiệp, công ty, cơ quan, trường học. Việc được truy cập từ xa cho phép đồng nghiệp , nhà cung cấp và khách hàng có thể giao tiếp với nhau một cách hiệu quả hơn, phá vỡ những rào cản về địa lí .

* Nhân viên ở xa nơi làm : Họ có thể truy cập máy tính của công ty ở từ xa và làm việc hiệu quả mọi lúc mọi nơi.
* Các phòng ban : Có thể chia sẻ, cộng tác, đào tạo và hỗ trợ nhau làm việc một cách hiệu quả hơn.

**Một số đánh giá của khách hàng về RealVNC :**

**Richard R :** “Chúng tôi đã dùng của phần mềm RealVNC® trong hơn 3 năm qua. Trong thời gian đó, nó đã cho phép chúng tôi kết nối từ xa với hơn 1000 máy tính để bàn và máy chủ để giải quyết sự cố, cài đặt phần mềm, máy in, v.v. giúp chúng tôi đã tiết kiệm được nhiều giờ lái xe và cho chúng tôi thời gian phản hồi nhanh hơn để giải quyết các vấn đề.

**Cavil Davis :** “Tôi đã thử nghiệm Teamviewer, Splashtop và Chrome Remote Desktop và nhận thấy rằng VNC® đánh bại tất cả chúng về cả chức năng và chi phí. Tôi có thể làm việc từ xa trên máy tính doanh nghiệp của mình và đôi khi quên rằng tôi đang sử dụng một máy tính từ xa vì tốc độ quá nhanh! ”

1. **Lịch sử :**

[Andy Harter](https://en.wikipedia.org/wiki/Andy_Harter) (Giám đốc điều hành của RealVNC Limited) và các thành viên khác của nhóm VNC ban đầu tại [AT&T đã](https://en.wikipedia.org/wiki/AT%26T) thành lập RealVNC Limited vào năm 2002. Một nhóm bộ phận khác của RealVNC đã tách ra thành một công ty riêng biệt ( [VNC Automotive](https://www.vncautomotive.com/) ) vào năm 2018.

1. **Tính năng chính :**

* **Điều khiển từ xa trực quan** : Sử dụng chuột và bàn phím hoặc màn hình cảm ứng. Gửi các phím Ctrl-Alt-Del và phím tắt, sao chép và dán văn bản qua lại.RealVNC Đáp ứng bất kì nền tảng, bàn phím hoặc ngôn ngữ nhập của bạn.
* **Hiệu suất** : RealVNC tối ưu hóa mạng của người dùng, khiến cho tốc điều khiển một cách nhanh chóng, như thể người dùng đang sử dụng máy tính của mình.
* **Hỗ trợ đa nền tảng** : RealVNC hỗ trợ đa nền đảng như các hệ điều hành Windows, Mac, Linux, Raspberry Pi, iOS và Android.
* **Truyền tệp, in và trò chuyện :** Trong một phiên, bạn không bị giới hạn chỉ tương tác với màn hình từ xa. Bạn có thể chuyển tệp qua lại, in tệp trực tiếp đến máy in cục bộ (thay vì máy in được gắn vào máy tính từ xa) và trò chuyện an toàn với những người dùng khác đã đăng nhập cùng lúc với bạn.
* **An toàn là trọng tâm của thiết kế** : VNC Connect được thiết kế từ đầu với tính bảo mật và cân bằng nhu cầu kiểm soát của bạn với các yêu cầu về quyền riêng tư theo quy định. Các phiên được mã hóa End-To-End bằng thuật toán AES 256-bit. Bảo vệ máy tính của bạn bằng xác thực đa yếu tố, đăng nhập một lần (SSO), kiểm soát truy cập chi tiết và các tùy chọn quyền mở rộng cho mỗi phiên.
* **Hỗ trợ đa ngôn ngữ** : Ngoài tiếng Anh, VNC Connect có sẵn bằng tiếng Pháp, Đức, Tây Ban Nha và Bồ Đào Nha Brazil. Các ngôn ngữ khác sẽ sớm được cung cấp.
* **Quản lý nhóm trực tuyến :** Một giao diện duy nhất để quản lý máy tính từ xa, những người dùng mà bạn mời vào nhóm của mình để chia sẻ quyền truy cập từ xa cũng như vai trò và quyền của họ.
* **Triển khai từ xa** : Đăng ký Enterprise cho phép bạn triển khai VNC Connect hàng loạt, ngoại tuyến hoặc từ xa bằng các công cụ kinh doanh tiêu chuẩn của bạn. Ngoài ra, bạn có thể định cấu hình VNC Connect từ xa thông qua chính sách hệ thống và khóa nó lại để tránh sửa đổi.
* **Máy tính để bàn ảo trong Linux** : Đăng ký Enterprise cho phép bạn tạo máy tính để bàn ảo trên máy tính Linux. Tính năng này có thể hữu ích, chẳng hạn để truy cập trực quan từ xa vào các hệ thống Linux không hiển thị hoặc cung cấp không gian làm việc riêng biệt cho những người dùng khác nhau chia sẻ tài nguyên.

## Chương 3 : Triển khai chương trình truy cập màn hình máy tính từ xa ứng dụng VNC :

* 1. **Các công cụ được sử dụng :**
* VMWare : Vmware là một phần mềm tạo ra máy ảo chạy trên các hệ điều hành windows, linux. Nó được cài đặt trực tiếp trên một server vật lý hay một chiếc máy tính cá nhân, việc cài trên máy tính cá nhân thì sẽ biến một chiếc máy tính cá nhân trở thành một server nhỏ.

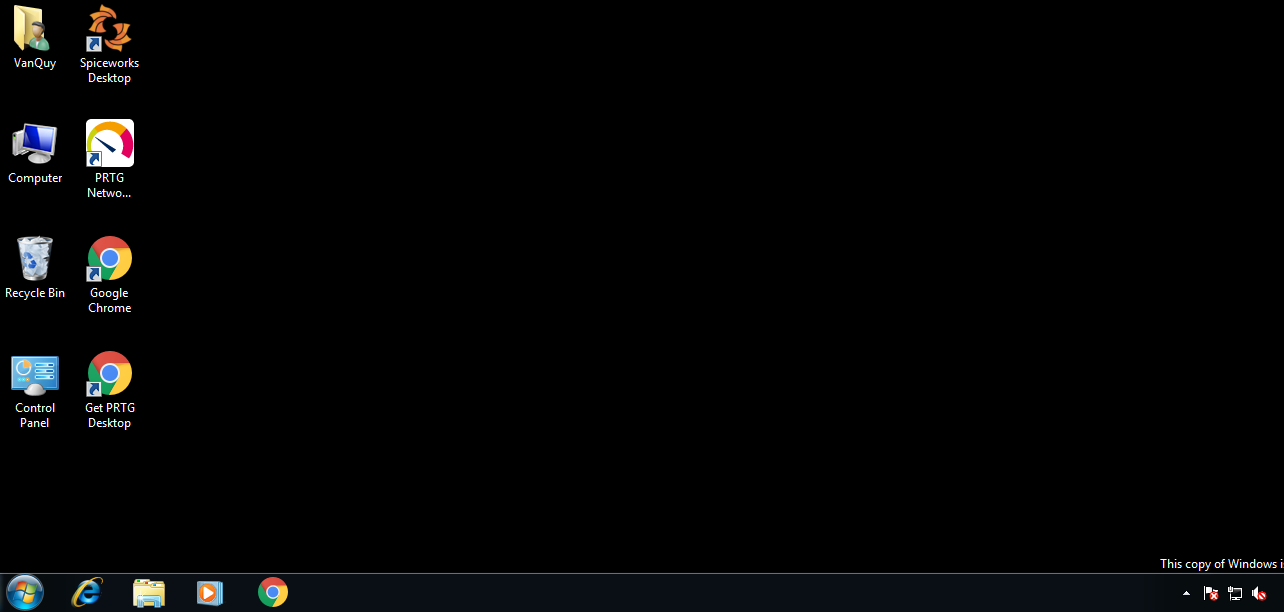


* Eclipse Editor : là một trình IDE dùng để lập trình Java (Eclipse cũng có thể dùng để lập trình các ngôn ngữ khác nhau C/C++, PHP… nhưng mạnh nhất là về Java). Mục đích để chạy mã nguồn mở Java trên công cụ này.

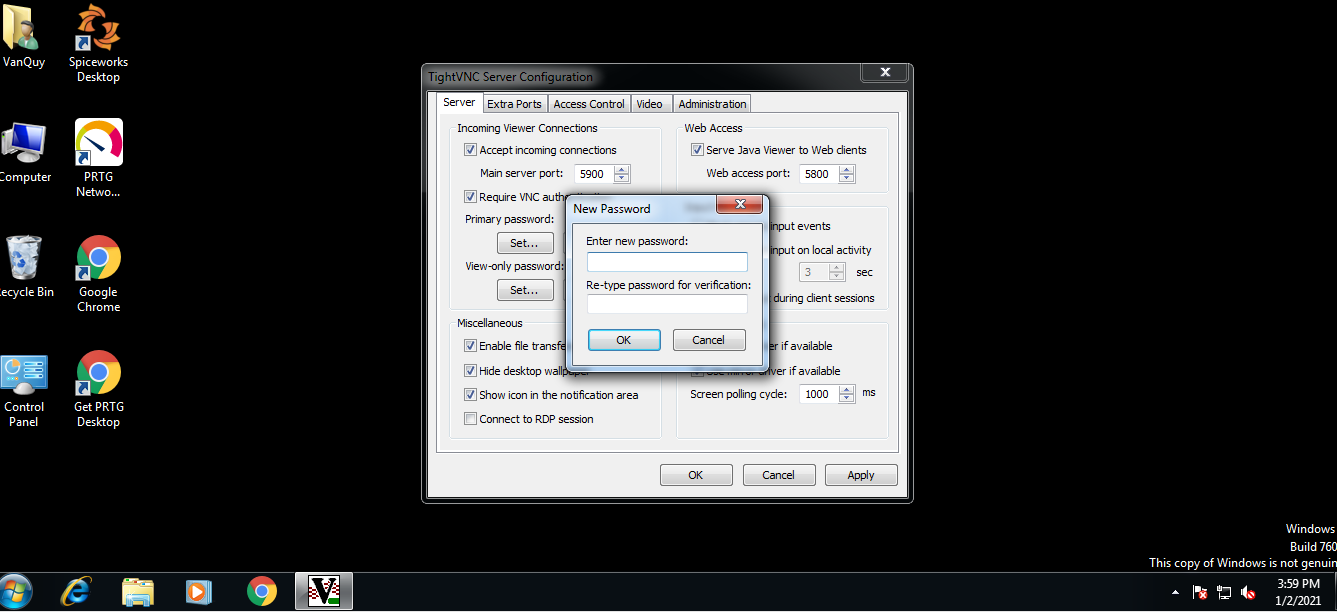


* 1. **Quá trình triển khai :**

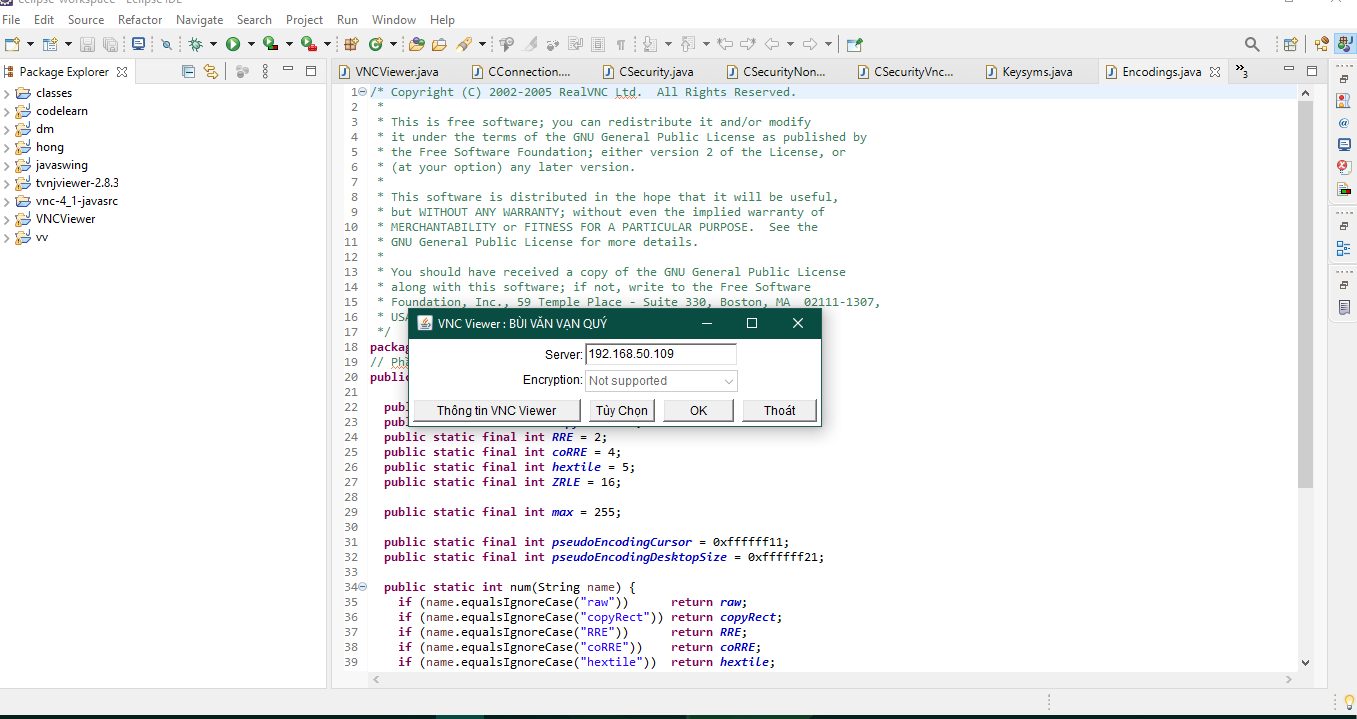
Bước 1 cài đặt hệ điều hành window 7 trên máy ảo VMWare :



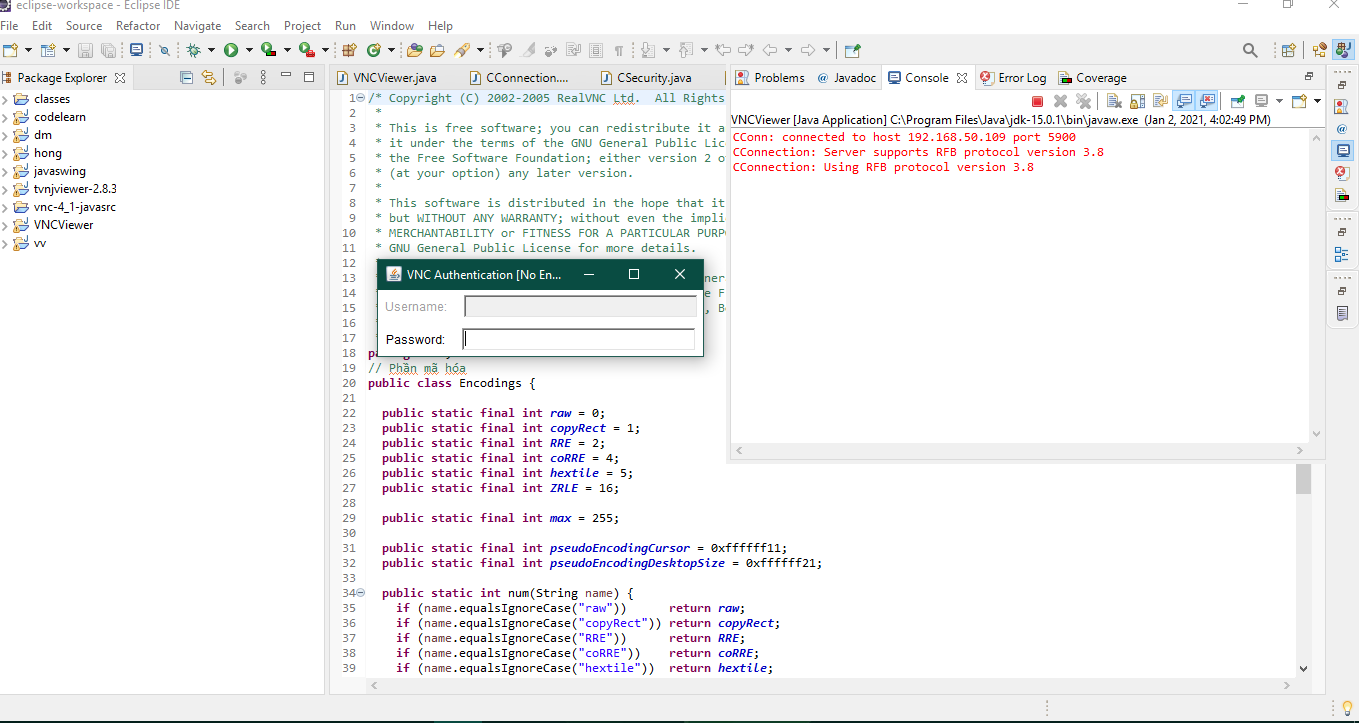
Bước 2 : Cài đặt VNC Server trên Window 7 và Set Password cho nó :



Bước 3 : Chạy VNC Client bằng Eclipse và kết nối máy chủ :

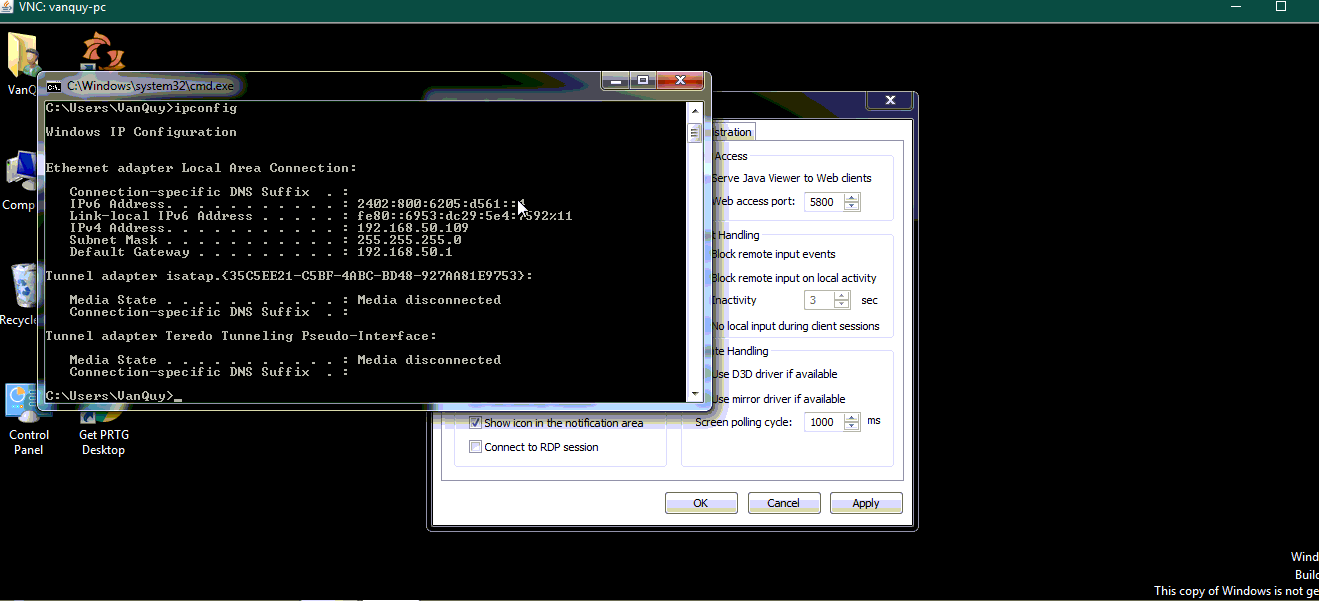


Trong quá trình này chúng ta sẽ nhập IP của máy chủ vào trong VNC Client .



Sau đó nhập Password mà chúng ta đã đặt ở trong bước 2 .

Kết quả nhận được :



# Phần III: Phụ Lục

## 1.Một số thuật ngữ :

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |