RESEARCH ARTICLE OPEN ACCESS

VNC Viewer Authentication and Security Using Remote Frame Buffer Protocol

D. Suguna Kuamari, B. Srinivasulu , G. Srinivas , G. Jaya Rao
[1,2,3,4] Department of Computer Science and Engineering,
St.Martin's Engineering College, Secundedrabad, Andhra Pradesh, India – 500014

Abstract
VNC is remote control software which allows you to view and fully interact with one computer desktop using a simple program on another computer desktop anywhere on the Internet. The two computers don't even have to be the same type, so for example you can use VNC to view a Windows Vista desktop at the office on a Linux or Mac computer at home. VNC is in widespread active use by many millions throughout industry, academia and privately. A proxy is used to send the image of the desktop to the cellular phone, to convert different devices, to suppress network traffics, and to support recovery from an unscheduled disconnection. A prototype of the proposed system has to be implemented using Java and will be tested on android phone. Virtual Network Computing is a simple protocol for remote access to graphical user interfaces. It is based on the concept of a Remote Frame Buffer (RFB). Several functions to the remote computer or the server desktop such as viewing the desktop, mouse click operations, sending a text message, opening of documents or files, opening of the task manager, manipulating of files and several other functions can be performed from our cellular phone.

Keywords: Virtual Network Computing, Remote Frame Buffer, Remote Control, Remote Desktop, Client-Server, Zooming-panning, Mouse Driver .

I. INTRODUCTION

Virtual Network Computing (VNC) is an ultra-thin client system based on a simple display protocol that is platform independent. VNC was originally developed at the Olivetti Research Laboratory in Cambridge, England. It achieves mobile computing without requiring the user to carry any hardware. RFB is a simple protocol for remote access to graphical user interfaces. Because it works at the framebuffer level it is applicable to all window-ing systems and applications, including X11, Windows and Macintosh. RFB is the protocol used in VNC.The remote endpoint where the user sits (i.e. the display plus keyboard and/or pointer) is called the RFB client or viewer. The endpoint where changes to the framebuffer originate (i.e. the windowing system and applications) is known as the RFB server. RFB is truly a "thin client" protocol. The emphasis in the design of the RFB protocol is to make very few requirements of the client. In this way, clients can run on the widest range of hardware, and the task of implementing a client is made as simple as possible. In computing, VNC is a graphical desktop sharing system which uses the protocol to control the computer remotely, but now according to our methodology remote computers can be accessed from cellular phones It transmits the keyboard and a mouse event from remote computer to cellular phones. The objective of our work is to develop an application through which user can monitor the computer over a network and can perform the administrative tasks from the cellular phones itself. The rest of the paper has been organized as follows: Section II proposes the novel approach and experimental results and Section III concludes the paper. In VNC system, server machines supplied not onlyapplications and data but also an entire desktop environment that can be accessed from any internet-connected machine using simple software VNC. VNC provide mobile computing without requiring the user to carry any device whatsoever. In addition, VNC allowed a single desktop to be accessed from several places simultaneously, thus supporting application sharing in the style of Computer-Supported Cooperative Work.

1.1 Purpose

VNC is remote control software which allows you to view and fully interact with one computer desktop using a simple program on another computer desktop anywhere on the Internet. The proposed system VNC Viewer lets you use your Android mobile device as a client for a VNC server so that we can view a desktop computer on a mobile phone. The tool is to be build up using Android SDK, which makes the application can be deployed on any android powered mobile phones.

1.2 Scope

One of the fastest growing industries now days are mobile industry. There are many competitors in this area who are doing research and development on new platforms & user experience. One such technology is Android from Google which is supported for Google phones. These phones are described as next Generation mobiles. There is no such application on mobile which acts as a VNC Client. We wanted to build an application on android powered mobile phone so that this phone acts as a client for a VNC Server.

1.3 Project Overview

The VNC Viewer lets you use your Android mobile device as a client for a VNC server so that we can view a desktop computer on a mobile phone. The tool is to be build up using Android SDK, which makes the application can be deployed on any android powered mobile phones.

1.4 Project Description

The Android SDK includes a comprehensive set of development tools. Requirements include Java Development Kit, the officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications. VNC is in widespread active use by many millions throughout industry, academia and privately. VNC Viewer lets you use your Android mobile device as a client for a VNC server. When we first open VNC Viewer, we should provide a connection configuration page. Here we can set up a connection with a VNC server (by providing IP address, port number, password), or choose an already configured connection. When this is done it should be able to handshake, authenticate and load the first frame.

II. SYSTEM ANALYSIS

Problem Definition

The user can access the data from the server system when he provided with the other system, so to avoid such problems, we have designed a VNC viewer so that the user can access the data from anywhere through mobile device.

2.1 Existing System

Till now there is no system that fits in the mobile phone to access the server system, the user can access the data only when is provided with another system.

2.2 Proposed System

We have a system that has out and out features to access the server system through mobile phone, when the system is provided with VNC server. The user can view the data in the server system from anywhere through mobile.

2.3 User Classes and Characteristics

End user of the application is the mobile phone user. User can connect to any desktop computer with the help of VNC server and view their desktop in mobile

2.4 Operating Environment:

Software Requirements used are Windows XP and any other latest editions. Java 1.6 and Android SDK. Hardware Requirements used are P4processor, 512MB of Main Memory (RAM) and 40GB hard disk and base memory.

2.5 Design and Implementation Constraints

All modules are coded thoroughly based on requirements from software organization. The software is designed in such a way that the user can easily interact with the screen. Software is designed in such a way that it can be extended to the real time business.

2.6 User Documentation

In our user manual we are going to keep the information regarding our product, which can be understandable by a new person who is going to use it. If a new person is using it, online help will be provided in that. We are going to explain each and every step clearly about our product so that any user can easily understand it.

2.7 Module Description

In this we have three modules

- Configuration
- RFB Implementation
- Activity GUI

Configuration

In this we specify the configuration of the VNC Server. This includes supplying IP Address, port number (5900 here for vnc by default) and password. We can choose from a list of already defined configurations. We use socket programming for server connection.

RFB Implementation

In this module we implement the Remote Frame Buffer protocol. This is used to access the remote computer windows GUI

Activity GUI.

In this module we develop all the activities needed to implement this project. This includes configuration screen, menu, canvas and settings.

III. PROPOSED METHODOLOGY

3.1 Display Protocol:

The display side of the protocol is based around a single graphics primitive: “put a rectangle of pixel data at a given x,y position” However , allowing various different encodings for the pixel data gives us a large degree of flexibility in how to trade off various parameters such as network bandwidth, client drawing speed and server processing speed. A sequence of these rectangles makes a framebuffer update (or simply update). An update represents a change from one valid framebuffer state to another, so in some ways is similar to a frame of video. The rectangles in an update are usually disjoint but this is not necessarily the case. The update protocol is demand-driven by the client. That is, an update is only sent from the server to the client in response to an explicit request from the client. This gives the protocol an adaptive quality. The slower the client and the network are, the lower the rate of updates becomes. With typical applications, changes to the same area of the frame buffer tend to happen soon after one another. With a slow client and/or network, transient states of the framebuffer can be ignored, resulting in less network traffic and less drawing for the client.

3.2 Input Protocol:

The input side of the protocol is based on a standard workstation model of a keyboard and multi-button pointing device. Input events are simply sent to the server by the client whenever the user presses a key or pointer button, or whenever the pointing device is moved. These input events can also be synthesized from other non-standard I/O devices. For example, a pen-based handwriting recognition engine might generate keyboard events.

3.3 Representation of pixel data:

Initial interaction between the RFB client and server involves a negotiation of the for-mat and encoding with which pixel data will be sent. This negotiation has been de-signed to make the job of the client as easy as possible. The bottom line is that the server must always be able to supply pixel data in the form the client wants. However if the client is able to cope equally with several different formats or encodings, it may choose one which is easier for the server to produce. Pixel format refers to the representation of individual colors by pixel values. The most common pixel formats are 24-bit or 16-bit “true color”, where bit-fields within the pixel value translate directly to red, green and blue intensities and 8-bit “colour map where an arbitrary mapping can be used to translate from pixel values to the RGB intensities. Encoding refers to how a rectangle of pixel data will be sent on the wire. Every rectangle of pixel data is prefixed by a header giving the X, Y position of the rectangle on the screen, the width and height of the rectangle, and an encoding type which specifies the encoding of the pixel data. The data itself then follows using the specified en coding.

3.4 Protocol Messages:

The RFB protocol can operate over any reliable transport, either byte-stream or message-based. Conventionally it is used over a TCP/IP connection. There are three stages to the protocol. First is the handshaking phase, the purpose of which is to agree upon the protocol version and the type of security to be used. The second stage is an initialization phase where the client and server exchange ClientInit and ServerInit messages. The final stage is the normal protocol interaction.

3.5 Security:

Once the protocol version has been decided, the server and client must agree on the type of security to be used on the connection. If the server listed at least one valid security type supported by the client, the client sends back a single byte indicating which security type is to be used on the connection: No. of bytes Type [Value] Description1 U8 security-type If number-of-security-types is zero, then for some reason the connection failed(e.g. the server cannot support the desired protocol version). This is followed by a string describing the reason (where a string is specified as a length followed by that many ASCII characters):No. of bytes Type [Value] Description4 U32 reason-length reason-length U8 array reason-string the server closes the connection after sending the reason-string. The security-type may only take the value 0, 1 or 2. A value of 0 means that the connection has failed and is followed by a string giving the reason, as described above. The security types defined in this document are: Number Name2 VNC Authentication. Other registered security types are:

Once the security-type has been decided, data specific to that security-type follows. At the end of the security handshaking phase, the protocol normally continues with the Security Result message. Note that after the security handshaking phase, it is possible that further protocol details over an encrypted or otherwise altered channel.

3.6 FramebufferUpdateRequest:

Notifies the server that the client is interested in the area of the frame buffer specified by x-position, y-position, width and height. The server usually responds to a FramebufferUpdateRequestby sending a FramebufferUpdate. Note however that a singleFramebufferUpdate may be sent in reply to several FramebufferUpdateRequests.The server assumes that the client keeps a copy of all parts of the frame buffer in which it is interested. This means that normally the server only needs to send incremental updates to the client. However, if for some reason the client has lost the contents of a particular area which it needs, then the client sends a Frame buffer Update Request with incremental set to zero(false). This requests that the server send the entire contents of the specified area as soon as possible. The area will not be updated using the CopyRect encoding. If the client has not lost any contents of the area in which it is interested, then it sends a FramebufferUpdateRequest with incremental set to non-zero (true). If and when there are changes to the specified area of the frame buffer, the server will send aFramebufferUpdate. Note that there may be an indefinite period between theFramebuffer Update Request and the FramebufferUpdate. In the case of a fast client, the client may want to regulate the rate at which it sends incremental FramebufferUpdateRequests to avoid hogging the network.

3.7 Extended system themes:

When it comes to the look-and-feel of the user interface, it's important to blend in nicely. Users are jarred by applications which contrast with the user interface they've come to expect. When designing your UIs, you should try and avoid rolling your own as much as possible. Instead, use a Theme. You can override or extend those parts of the theme that you need to, but at least you're starting from the same UI base as all the other applications. For all the details.

3.7.1 Design your UI to work with multiple screen resolutions:

Different Android-powered devices will support different screen resolutions. Some will even be able to change resolutions on the fly, such as by switching to landscape mode. It's important to make sure your layouts and drawables are flexible enough to display properly on a variety of device screens. Fortunately, this is very easy to do. In brief, what you must do is provide different versions of your artwork (if you use any) for the key resolutions, and then design your layout to accommodate various dimensions. (For example, avoid using hard-coded positions and instead use relative layouts.) If you do that much, the system handles the rest, and your application looks great on any device.

3.7.2 Assume the network is slow:

Android devices will come with a variety of network-connectivity options. All will have some data-access provision, though some will be faster than others. The lowest common denominator, however, is GPRS, the non-3G data service for GSM networks. Even 3G-capable devices will spend lots of time on non-3G networks, so slow networks will remain a reality for quite a long time to come. That's why you should always code your applications to minimize network accesses and bandwidth. You can't assume the network is fast, so you should always plan for it to be slow. If your users happen to be on faster networks, then that's great ,their experience will only improve. You want to avoid the inverse case though: applications that are usable some of the time, but frustratingly slow the rest based on where the user is at any given moment are likely to be unpopular. One potential gotcha here is that it's very easy to fall into this trap if you're using the emulator, since the emulator uses your desktop computer's network connection. That's almost guaranteed to be much faster than a cell network, so you'll want to change the settings on the emulator that simulate slower network speeds. You can do this in Eclipse, in the "Emulator Settings" tab of your launch configuration or via a command line option when starting the emulator.

3.7.3 Don't assume touch screen or key board:

Android will support a variety of handset form-factors. That's a fancy way of saying that some Android devices will have full "QWERTY" keyboards, while others will have 40-key, 12-key, or even other key configurations. Similarly, some devices will have touch-screens, but many won't. When building your applications, keep that in mind. Don't make assumptions about specific keyboard layouts -- unless, of course, you're really interested in restricting your application so that it can only be used on those devices.

3.7.4 Do converse the device battery:

A mobile device isn't very mobile if it's constantly plugged into the wall. Mobile devices are battery-powered, and the longer we can make that battery last on a charge, the happier everyone is — especially the user. Two of the biggest consumers of battery power are the processor, and the radio; that's why it's important to write your applications to do as little work as possible, and use the network as infrequently as possible. Minimizing the amount of processor time your application uses really comes down to writing efficient code writing efficient code. To minimize the power drain from using the radio, be sure to handle error conditions gracefully, and only fetch what you need. For example, don't constantly retry a network operation if one failed. If it failed once, it's likely because the user has no reception, so it's probably going to fail again if you try right away; all you'll do is waste battery power. Users are pretty smart: if your program is power-hungry, you can count on them noticing. The only thing you can be sure of at that point is that your program won't stay installed very long.

3.7.5 No Pan-Trackball Mouse mode is only available in Fit to Screen scaling and is the only input mode available then. In this mode the touch screen is not used. Keyboard events are sent to the server and the trackball (if the user's device, like the G1, has a trackball) controls the VNC mouse. The figure 2.7 shows the screenshot of no pan; trackball mode.

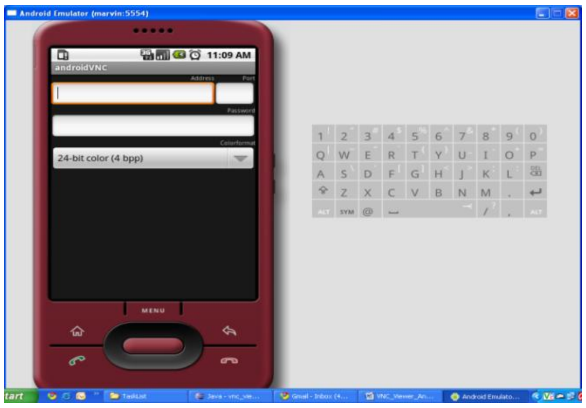
Fig : No pan, Trackball mouse mode

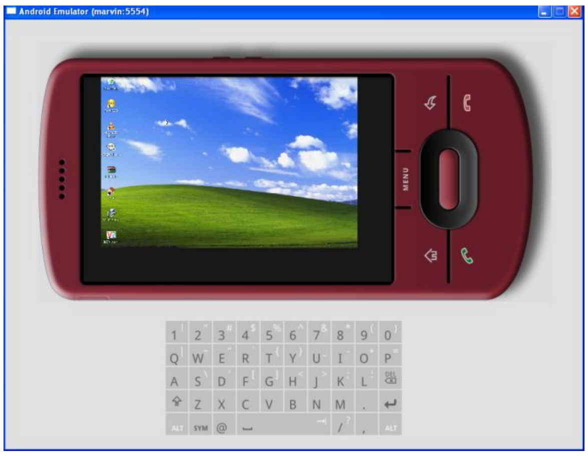
Touch Pan; Trackball Mouse drags on the touch screen to pan the device display over the VNC display. Keyboard events are sent to the server. The trackball controls the VNC mouse. Pressing the trackball sends a mouse click; holding the ball down while rolling accomplishes a click and drag. This is the default input mode when scaling is set to One-to-

One.

DPad Pan; Touch Mouse Mode uses the directional pad (available on some devices) to pan the display over the VNC display. Touch the screen to send a mouse click; touch and slide to send a mouse drag. Use the camera button while touching the screen to simulate a right-button click or drag.

SCREEN SHOTS





REFERENCES

[1] Md. Sanaullah Baig, Rajasekar M. and Balaji P "Virtual Network Computing Based Remote Desktop Access", International Journal of Computer Science and Telecommunications, Vol. 3, No. 5, 2012.

[2] Margaret Butler, "Android: Changing the Mobile Landscape", IEEE Pervasive Computing.

[3] Ha-Young Ko, Jae-Hyeok Lee, Jong-Ok Kim, "Implementation and Evaluation of Fast Mobile VNC Systems", IEEE Transactions on Consumer Electronics, Vol. 58, No. 4, 2012.

[4] Daniel Thommes, "RemoteUI: A High-Performance Remote User Interface System for Mobile Consumer Electronic Devices", IEEE Transactions on Consumer Electronics, Vol. 58, No. 3, 2012.

[5] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood and Andy Hopper, "Virtual Network Computing", IEEE Internet Computing.

[6] The RFB Protocol ,Tristan Richardson , Real VNC Ltd (formerly of Olivetti Research Ltd / AT&T Labs Cambridge) Version 3.8

BIOGRAPHY

Mrs D.Suguna Kuamari, Post Graduated in Computer Science (M.Tech), ANU, 2010, and Graduated in Information Technology (B.Tech) From JNTU Hyderabad, 2006. She is working presently as Associative Professor in Department of Computer Science & Engineering in **St. Martin's Engineering College**, RR Dist, A.P, INDIA. She has 5+ years Experience. Her Research Interests Include Software Engineering, Cloud Computing, Operating Systems and Information Security.

Mr. B.Srinivasulu, Post Graduated in Computer Science (**M.Tech**) From **JNTU**, Hyd in 2010 and Graduated in Information Technology (B.Tech) from JNTUH, in 2008. He is working as Assistant Professor in Department of Computer Science & Engineering in **St.Martin's Engineering College**, R.R Dist, AP, India. He has 3+ years of Teaching Experience. His Research Interests Include Network Security & Data Warehousing and Data Mining.

Mr. Gajula Srinivas, Post Graduated in Computer Science (**M.Tech**) From **JNTU**, Hyd in 2011 and Graduated in Information Technology (B.Tech) from **KAKATIYA UNIVERSITY**, Warangal, 2006. He is working as Assistant Professor in Department of Computer Science & Engineering in **Visvesvaraya College of Engineering & Technology**, R.R Dist, AP, India. He has 4 years of Teaching Experience. His Research Interests Include Network Security & Data Warehousing and Data Mining.

Mr Gudeme Jaya Rao, Post Graduated in Computer Engineering (M.Tech) From VTU, Karnataka 2009, and graduated in Computer Science & Information Technology (B.Tech)from JNTU Hyderabad, 2002. He is working presently as Associate Professor in Department of Computer Science & Engineering in **St.Martin's Engineering College**, RR Dist, A.P, INDIA. He is has 9+ years Experience. His Research Interests Include Software Engineering, Network Security & Cloud Computing.