Software Requirements Specification:
Customisable VNC Viewers

Brandon Byskov
1068517
byskovbm@mcmaster.ca

Xuchao Ding
1233855
dingx3@mcmaster.ca

Natalie Perna
1066785
pernanm@mcmaster.ca

October 10th, 2014

Capstone Project
Department of Computing and Software
McMaster University

Supervised by:
Dr. Wolfram Kahl
kahl@mcmaster.ca

Instructed by:
Dr. Rong Zheng
rzheng@mcmaster.ca

Revision History

Date	Version	Comments
2014-10-10	0.1	First draft.

Acknowledgments

This document is based upon the IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998).

Contents

1   Introduction

1.1   Purpose

The purpose of this software requirements specification is to outline the requirements of the software that will be built for the 4th year capstone project by the group consisting of Brandon Byskov, Xuchao Ding, and Natalie Perna. This document is intended to be used by the development team, supervisors, and course instructors.

1.2   Scope

This document will outline the software requirements of the *Customisable VNC Viewer*. This software will allow a user to an run application or desktop on a remote server, while viewing that application or desktop in a local window and interacting using local input devices. Furthermore, the user will be allowed to segment contents of the application window across multiple physical displays. This would allow a user to tile multiple physical displays beside each other and view a software application that can span continuously across all of the displays. The Customisable VNC Viewer will be written in code that can be easily reused for other projects. There is a goal to have this software implemented on McMaster University Computing and Software Department's VidaLab large display.

1.3 Definitions, acronyms, and abbreviations

In general, the definitions of terms used in this document conform to the definitions provided in the RFB Protocol.[1]

The definitions below are key terms as they are used in this specification.

1.3.1 Virtual Network Computing (VNC)

Virtual Network Computing (VNC) is a system for graphical desktop sharing.

1.3.2 VNC Server

The VNC server, or RFB server, is the remote endpoint where the applications and graphical desktop environment are running, and changes to the framebuffer originate. It may also be referred to as the host.

1.3.3 VNC Client

The VNC client, or RFB client, is the user's endpoint, where the display is projected (i.e. onto a monitor), and input devices are connected (i.e. keyboard, mouse, etc.). It may also be referred to as the viewer.

1.3.4 RFB Protocol

The remote framebuffer (RFB) Protocol specifies the rules and procedure for remote access to a graphical user interface. RFB is the protocol used in VNC.

1.4 References

[1] T. Richardson and J. Levine. (2011, Mar.) The remote framebuffer protocol. RealVNC Ltd. [Online]. Available: http://tools.ietf.org/html/rfc6143

[2] VNC® Open. RealVNC Ltd. [Online]. Available: https://www.realvnc.com/products/open/

1.5 Overview

The following document contains detailed non-techincal requirements of the Customisable VNC Viewer. Section 2 describes the software product broadly, including descriptions of its functions, users, and constraints. Section 3 describes in greater detail the specific requirements of this software. This includes detail about the external interfaces, functional and performance requirements, constraints, and attributes.

2 Overall description

2.1 Product perspective

The customisable VNC software will extend the functionality of existing VNC software. It must have two application parts: a VNC client and VNC server. The two parts will interact with each other using a reliable networking protocol. The VNC client must interact with the operating system of the client machine to recieve input device events. It must also interact witht the window system of the client machine to display graphical images to the user. It will have a graphical users interface to allow the user to connect the client to the server. The VNC server must interface with the server's virtual machine environment. Both the VNC client and server will run on a Linux operating system, witht he specific operating system version to be determined.
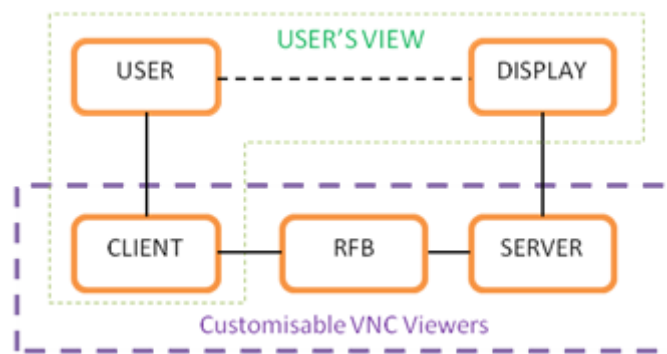

Figure 1: Structure of remote control system.

Figure 1 show the layout of the software system. The VNC server will communicate over a network using the RFB protocol to the client. The server will generate a virtual display that it sends to the client to be viewed by the user. The user will only physically interact with the client to view that display and interact with it, but will see the applicaiton of the client as it is generated on the server.

This product perspective is not complete, and will be expanded as the requirements are better determined.

2.2 Product functions

The Customisable VNC Viewer should be able to connect a client to a remote server. The client will send initialization parameters to the server. The client will update the display with the graphical image data sent by the server. The user controls the application by keyboard and mouse events. The client will be able to choose to recieve only a portion of the total virtual display from the server.

2.3 User characteristics

There would be three kinds of users:

The client users of this software would be those who could benefit from tiling multiple client displays to a single server to create a large, high-resolution display.

The server managers will be those users who must maintain the software and virtual environments on the VNC server.

Software developers potential users who might reuse the software codebase that is developed by this project.

2.4 Constraints

2.4.1 Backwards Compatibility

The software should be compatible with existing VNC clients and servers and the RFB protocol. The software should introduce new capabilities, but be backwards compatible and not break any existing functionality.

2.4.2 Hardware limitations

There must be networking connectivity between the client and server. The server must be capable of accepting and managing the connections of multiple clients simultaneously. There must be sufficient network bandwidth to allow communication between client and server.

2.4.3 Interface to other applications

The VNC software must interface with the window system of the client, and interface with the virtual environment of the sever.

2.4.4 Parallel operation

The server must allow parallel communication to all of its clients. This would allow each client to send input device commication to the server in parallel.

2.4.5 Control functions

The client must allow control by mouse and keyboard operations.

2.4.6 Criticality of the application

The client must be allowed to disconnect or quit the session without failure at the server.

2.4.7 Safety and security considerations

Unauthenticated information should not be viewed or modified when the client connects using ==and== authenticated connection.

2.5 Assumptions and dependencies

This project builds on existing VNC server and client software. The software will run in a Linux environment. The software will be tested to work with McMaster University Computing and Software Department's VidaLab large display. Any changes to the dependencies and assumptions would affect the requirements.

3 Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.1.1 Connection initiation The first dialog presented to the user upon starting the client application should allow the user to enter a URL or IP address of the server to connect to, and a button to start the connection. There should be some text feedback indicating that a connection is being made.

3.1.1.2 Connection failure If a VNC server is not detected at the specified connection string, or the address cannot be resolved, then an error message indicating the error should be displayed to the user in the dialog box.

3.1.1.3 Authentication If the server requires a password for authentication, then the user should be prompted within the dialog box to enter a password.

3.1.1.4 Authentication failure If the connection fails due to a rejected authentication from the server, then the user should be notified in the dialog box via an error message that the password was incorrect.

3.1.1.5 Connection and handshake During the connection and handshake step, the dialog box should give the user feedback that the connection setup is in progress.

3.1.2 Hardware interfaces

Not applicable.

3.1.3 Software interfaces

The client and server applications should run on current Linux distributions. (Note: This may later be expanded to include more Linux distributions and/or Mac OS X depending on the ease of code portability.) More specific system requirements to be determined in later document versions.

3.1.4 Communications interfaces

VNC-specific communication will take place via the RFB protocol/interface. Communication will occur over ports 22 and 5900, so both should be open to the network.

3.2 Functional requirements

3.2.1 Requesting connection

The system will accept a server (host) IP address or URL from the user. The user will be allowed to start the connection after the connection string has been entered. If a VNC server can be reached at the specified endpoint, then the challenge-response step will begin. If a VNC server cannot be reached at the specified endpoint, then the user will be given a descriptive feedback message, and allowed to modify the connection string before optionally retrying.

3.2.2 Challenge-response scheme

If the client successfully initiates a connection with the server, then server may request authentication from the client via a challenge-response scheme. Our client will support unauthenticated, and password-authenticated schemes, in accordance with VNC proper. If unauthenticated connections are allowed, the client and server will proceed to the handshake phase. If a password is required for connection, the user will be prompted to enter a password. If the server authenticates the password, the handshake will begin. If the server rejects the password, the connection will be cancelled and the user will be allowed to modify the connection string before optionally retrying.

3.2.3 Client-server handshake

After requesting a connection, the VNC client will negotiate capabilities with the VNC server. The client and server will exchange messages to determine agreed upon: character encoding, desktop size, and pixel format. It must also be negotiated that both the client and server support customisable display segmentation (i.e. both VNC client and server are our forked versions).

3.2.4 Display segmentation negotiation

Having negotiated the total desktop size with the server in the handshake step, the client will now prompt the user with the total dimensions and allow the use to enter the requested segment in the form of two (x, y) coordinate pairs: bottom-left and top-right, where the bottom-left of the total server desktop is $(0,0)$ and the top-right is the (`horizontal resolution`, `vertical resolution`).

3.2.5 Session start

After submitting the requested segment of the total desktop, the client will display the segment in a viewer port window, updating the screen periodically with the server framebuffer stream. According the RFB protocol, the display side should be based on a single graphics primitive: "Put a rectangle of pixel data at a given x, y position.". The first display on the client will be sent from the server in this format.

3.2.6 Adaptive update

The set of rectangles that comprises the pixel data for the segment being viewed by the client must be regularly updated with a framebuffer update. VNC proper already supports transmitting only the changed portion of the framebuffer state. Our server fork will have the additional capability of only transmitting the changed segments which are within the segment requested by the client over the network. Updates will be demand-driven, that is, they will only be sent by the server after a request for an update is received from the client; this is done in accordance with the RFB protocol. The server will coalesce all updates since the clients previous request before sending. This creates an effect of adaptive updates, maximizing framebuffer update frequency according to network speed.

3.2.7 User input

The client will accept user input in the form of keyboard and mouse (multibutton pointing device). The client will transmit all input events (key presses, mouse movements and mouse clicks) to the server. The display will respond as if the inputs had occurred on the machine running the applications and display environment.

3.2.8 Disconnection

If the client is disconnected from the server for any reason (lost client network connection, lost server network connection, server shutdown, etc.), the client will display a message to the user

3.2.9 Client quitting

When the client application is quit, either through an application menu, system force quit, or system shutdown of the client machine, no further action is necessary. The server need not be notified.

3.3 Performance requirements

Performace requirements must be relative to existing solutions. The client should have a proportionally a higher framebuffer refresh rate for displaying a segment of the screen when compared to a standard client displaying the entire screen. Specific requirements to be determined in a later version of this document.

3.4 Design constraints

The software is constrained to adhere to the VNC specification and all requirements of the RFB protocol. The client and server implementations should be compatible with standard VNC proper systems as well, i.e. not supporting our additional features.

3.5 Software system attributes

3.5.1 Availability

To be determined.

3.5.2 Interoperability

Our VNC client should be capable of interoperating with any proper VNC server, by negotiating shared capabilities and adhering to the RFB protocol.

Likewise, any VNC client should be capable of interoperating with our VNC server.

The platform on which the client or server with which our system is connecting should not be relevant to the system function. (For example, although we will not directly support a Microsoft Windows application, existing VNC clients for Windows should be able to connect to our VNC server, albeit, without the added features only supported by our client and server.)

3.5.3 Maintainability

This system will be written with idiomatic, clear, modular code that is well-tested and specified to increase readability and maintainability.

3.5.4   Portability

This software should run on any current Linux operating system. More specific requirements to be determined.

3.5.5   Reusability

Along with the applications, we will also produce well-documented Haskell libraries with clean interfaces to serve as basis for other projects.

3.6   Other requirements

3.6.1   Licensing

Because our system will be based on the original open-source source code for the VNC project, our software will necessarily be licensed under the conditions of the GNU General Public License.[2]

Appendices

Index