Virtual Network Computing

II stage Project Report
under
Prof G. Sivakumar

by
Amit Chandel 01005001
Nilesh Bansal 01005006
Suddha Kalyan Basu 01005008
Abhisekh Shankaran 01D05009

What is VNC? - A practical introduction

VNC stands for ***Virtual Network Computing***. It is, in essence, a remote display system which allows you to view a computing 'desktop' environment not only on the machine where it is running, but from anywhere on the Internet and from a wide variety of machine architectures.

VNC – How it works

VNC is based on a "*thin-client*" protocol: it has been designed to make very few requirements of the viewer. 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.

VNC Protocol

The VNC protocol is a simple protocol for remote access to graphical user interfaces. It is based on the concept of a ***"remote framebuffer"*** or RFB.

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 . 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.

The simplest encoding type is ***raw pixel data***. In this case the data consists of n pixel values where n is the width times the height of the rectangle. In this case the data consists of n pixel values where n is the width times the height of the rectangle.

The ***copy rectangle encoding*** is a very simple and efficient encoding which can be used when the client already has the same pixel data elsewhere in its framebuffer. The encoding on the wire simply consists of an X,Y coordinate. This gives a position in the framebuffer from which the client can copy the rectangle of pixel data. This can be used when the user moves a window across the screen.

RRE stands for ***rise-and-run-length encoding*** and is aimed at situations where compression is desired but the RFB client is insufficiently powerful to perform any decompression fast enough to maintain interactive performance. The basic idea behind RRE is the partitioning of a rectangle of pixel data into rectangular subregions (subrectangles) each of which consists of pixels of a single value and the union of which comprises the original rectangular region.

CoRRE (***compressed RRE***) is a variant of RRE, where we guarantee that the largest rectangle sent is no more than 255x255 pixels. For a typical desktop, this results in better compression than RRE.

Hextile is a variation on the CoRRE idea, position and size of each tile do not have to be explicitly specified - the encoded contents of the tiles simply follow one another in the predetermined order.

Adaptive Update Protocol

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, but it is usually only a small area of the framebuffer that will be affected by a given update. Each rectangle may be encoded using a different scheme. The server can therefore choose the best encoding for the particular screen content being transmitted and the network bandwidth available. The update protocol is demand-driven by the client. That is, an update is only sent by the server 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. Each update incorporates all the changes to the 'screen' since the last client request. With a slow client and/or network, transient states of the framebuffer are ignored, resulting in reduced network traffic and less drawing for the client. This also improves the apparent response speed.

Connection Setup and Shutdown

When the connection between a client and a server is first established, the server begins by requesting authentication from the client using a challenge-response scheme, which typically results in the user being prompted for a password at the client end. The server and client then exchange messages to negotiate desktop size, pixel format, and the encoding schemes to be used. The client then requests an update for the entire screen, and the session begins. Because of the stateless nature of the client, either side can close the connection at any time without adverse consequences.

Initial Handshaking Messages

Following constitutes initial messages :

1. Protocol Version
2. Authentication
3. Client Initialisation
4. Server Initialisation

Client to Server messages

1. Set Pixel Format
2. Fix Colour Map Entries
3. Set Encodings
4. Framebuffer Update Request
5. Key Event
6. Pointer Event
7. Client Cut Text

Server to Client messages

1.FramebufferUpdate
2.SetColourMapEntries
3.Bell
4.ServerCutText

VNC consists of 5 programs

- **vncviewer** - this is the VNC viewer, or client, program for X.
- **vncserver** - this is a wrapper script which makes starting an X VNC server (i.e. desktop) more convenient. It is written in Perl, so to use the script you need that.
- **vncpasswd** - this program allows you to change the password used to access your X VNC desktops. The vncserver script uses this program when you first start a VNC server.
- **vncconnect** - this program tells a running instance of Xvnc to connect to a listening VNC viewer (normally the connection is made the other way round i.e. the viewer connects to Xvnc).
- **Xvnc** - this is the X VNC server - it is both an X server and a VNC server. You normally use the vncserver script to start Xvnc.

What we plan to do.

In the existing code for VNC there are a few bugs .They are listed as :-

1. The keyboard keys no longer work properly as the key functions gets jumbled up. (for eg, pressing key 'M' shows no response while pressing '/' functions as the enter key).
2. Once the full screen mode is enabled it is not possible to come back to your original desktop(unless cumbersome things like pressing ALT F1,doing logout and then cancelling it is done)
3. At present, exporting of sound is not possible.
4. Sometimes , opening up of xterm,console shows an unusually wide size of the window.
5. When we logout of the remote desktop(eg KDE) then we reach a black screen , and nothing can be done then. So we plan to find some solution to the problem.

In addition to fixing above bugs we will also try to

1. Make VNC more secure.
2. Find out some way by which 2 comps with some proxy server in between can also be connected by VNC.
3. Make it more user friendly.