SNAPWebD:

a web desktop for Standalone desktop or Nfs server using APache (SNAP) web server

A Dissertation Submitted in partial fulfillment of the degree of

Master of Technology
in
Computer Science

By
S S Krishna Vutukuri

School of Computer and Information Sciences
University of Hyderabad, Gachibowli
Hyderabad - 500046, India

June, 2013

CERTIFICATE

This is to certify that the dissertation entitled "**SNAPWebD**" submitted by **S S Krishna Vutukuri** bearing Reg. No. 11MCMT35, in partial fulfillment of the requirements for the award of Master of Technology in Computer Science, is a bona fide work carried out by him under our supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. P. Anupama
School of Computer and Information Sciences,
University of Hyderabad

Dr. S. Durga Bhavani
School of Computer and Information Sciences,
University of Hyderabad

Dr. T. Sobha Rani
School of Computer and Information Sciences,
University of Hyderabad

Dean,
School of Computer and Information Sciences,
University of Hyderabad

DECLARATION

I, **S S Krishna Vutukuri** hereby declare that this dissertation entitled "**SNAPWebD**" submitted by me under the guidance and supervision of Dr.Anupama P, Dr.Durga Bhavani S and Dr.Sobha Rani T is a bona fide work. I also declare that it has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Date:

S S Krishna Vutukuri

Reg. No.: 11MCMT35

Signature of the Student

Paper Submitted

- Submitted a paper titled **"Web Desktop for Users of NFS Logical Volumes"** in *IEEE International Symposium on Web Systems Evolution*, Sep. 2013.

This paper implements a web desktop for a user to access his/her home directories of NFS server as well as a standalone system. We presented our entire work in this paper.

Acknowledgments

I would like to express my grateful honor to my supervisors **Dr.Anupama Potluri, Dr.Dhurga Bhavani S and Dr.Sobha Rani T** who guided me to complete this project successfully. They really helped me a lot to finish my project well before the time and also personally to improve my communication skills. They all supported me very well to improve my health when my health did not support me to do the project. I started my project very late and finished it very early only because of my supervisors' encouragement.

I am so glad to get suggestions from **Prof. Chakravarthi Bhagavathi** to extend my project features.

I am so grateful to my Dean of School **Prof. Arun K Pujari** for providing a good computing facility and good lab environment.

I am very thankful to my friends who helped me many times in many critical situations.

S S Krishna Vutukuri

Abstract

The web browser is the most ubiquitous software today on all kinds of devices including PCs, tablets and mobile phones. At the same time, access to high bandwidth network services from all these devices has become routine with the advent of 3G/4G mobile services and WiFi hot spots all over the world. This has lead to the development of cloud computing where services such as applications or desktops can be accessed from remote servers through the Internet.

In this dissertation, we present our work in developing a Web Desktop for the typical environments found in many corporates and educational institutions. These environments consist of NFS mounted logical volumes of home directories of users authenticated using LDAP and NFS application servers that are transparently mounted based on access to a particular piece of software. The Web Desktop, which is a free and open source software, will allow users of such organizations to access their home directories and remote applications through the web browser. The web server transparently mounts the NFS partitions as needed and provides access to the files and applications of the organization to the users through the web browser.

The advantage of this tool is that the client system does not need any additional software to be installed. The fact that the firewalls of most organizations allow web access and none other is also a motivation for the development of the web desktop. No additional ports need to be opened up in the firewall mitigating the chances of attacks through security loopholes in other softwares.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In most institutions today, the users' access to the IT infrastructure is deployed in two ways – either using the individual desktops with a few shared folders as in typical Windows deployments or NFS mounted logical volumes accessed through thick or thin clients in Unix/Linux environments. With Unix/Linux deployments using NFS, the user can remotely access their home account using Virtual Networking Connection (VNC) over Virtual Private Networks (VPNs) of their organizations. This requires installation and configuration of the VPN and VNC software on both the client and server systems. The VPN configurations are typically quite complex. VNC uses a protocol which requires the firewalls of the organizations to open up this port.

A further development has been the use of Virtual Machines (VMs) with distributed file systems such as Hadoop. VMware has been at the forefront of deployment of VMs in IT infrastructure of companies. It has also built remote desktop products to work with its virtualization software. These are costly solutions that require an organization to move away from its already existing infrastructure. It also requires extensive IT support to work.

The Web Browser is an ubiquitous piece of software on all devices of communication today including PCs, Laptops, Tablets, Mobile phones etc.. This, combined with the availability of high bandwidth network access, has spurred the growth of *cloud computing*. Cloud computing is centralized transparent access to various services in the network anywhere, anytime. One of the services that is gaining a lot of momentum is a desktop environment or an operating system through the browser called Web Desktop and WebOS respectively. This extends independence from hardware and specific operating systems for users and helps portability. There are many products being offered to use the web browser as the desktop environment.

1.1 Motivation

Most of the products in the market today are limited to special accounts created specifically to host files/directories that a user wishes to access from the browser at any time from anywhere. These products do not allow access to the organization's existing infrastructure through the web browser. Many of the feature-rich products are proprietary and commercial involving a lot of cost to deploy them. Many of them also require installing additional software on client systems. They use protocols such as RDP or ICA to achieve remote access to the users' repositories. This might involve opening other ports in the firewalls of the organization and make them liable to attacks. Our product addresses these lacunae in the remote desktop products.

1.2 SNAPWebD: Proposed Web Desktop

We developed a Web Desktop, **SNAPWebD**, that we believe will be ideal for small offices and educational institutions, to provide access to the users' accounts of these organizations. Most of these organizations use NFS logical volumes with LDAP user authentication. Our **SNAPWebD** allows such NFS logical volumes to be accessed via a browser when our software is installed on the web server.

Differentiating Features: Our **SNAPWebD** has the following features that differentiate it from other existing remote/web desktop products:

- It does not require installation of any additional software on the client system.
- It is an open source and free software. Hence it provides economical access to users home accounts for small offices and educational institutions.
- It is a lightweight software. In that, it uses simple JSP and JQuery software on the server side to accomplish all the tasks.
- The IT challenge in implementing it is minimal as it uses the typical computing environment in most institutions today and requires only the installation of the JSP and JQuery software on the web server.
- It does not require additional ports to be opened up in the firewall, thus mitigating the vulnerability to attacks.
- Ability to access the home directory of a user on any arbitrary machine given its IP address or name and user-name, password of the user account in machine.

1.3 Organization of the Dissertation

The rest of the dissertation is organized as follows: we review some of the existing web desktop and other related products in Chapter 2. The architecture and implementation of the **SNAPWebD** are given in Chapter 3. I have also given a few use case scenarios in Chapter 3. We conclude with Chapter 4 listing the advantages of our **SNAPWebD** over other similar products and the enhancements planned for the future.

Chapter 2

Related Work

Desktop Virtualization is classified into 2 categories.

- Virtual Desktop
- Web Desktop

2.1 Virtual Desktop vs Web Desktop

Virtual Desktop is a technology that can access a system desktop environment or an application server remotely. This technology uses both application virtualization and user profiles management(files and directories) to provide a complete desktop virtualization. However this requires special client software on local system, e.g., Virtual Network Connectivity (VNC).

Web Desktop is different from Virtual Desktop in that it does not require special client software. Instead, it uses the ubiquitous web browser. This allows the desktop to be accessed from any device such as a tablet, mobile, laptop, desktop etc., which have a browser. In this the data, files, applications and processes reside on the remote system and are displayed in the browser when a user logs in or connects to the remote server. This helps in allowing access to the desktop without any further configuration of firewalls. Examples of Web Desktop are Open Virtual Desktop (OVD), XenDesktop, VirtualBox, Oracle Secure Global Desktop, appFlower, *eyeOS*, etc..

In the rest of this chapter we review the following software:

- Open Virtual Desktop(OVD)
- Lively Kernel

- EyeOS
- Remote Desktop on thin clients for Virtual Machines

2.2 Open Virtual Desktop (OVD) [7]

This is an open source product. It delivers both applications and files to end users. It works for both Linux and Windows Operating Systems. The architecture of OVD is shown in Fig. 2.1.

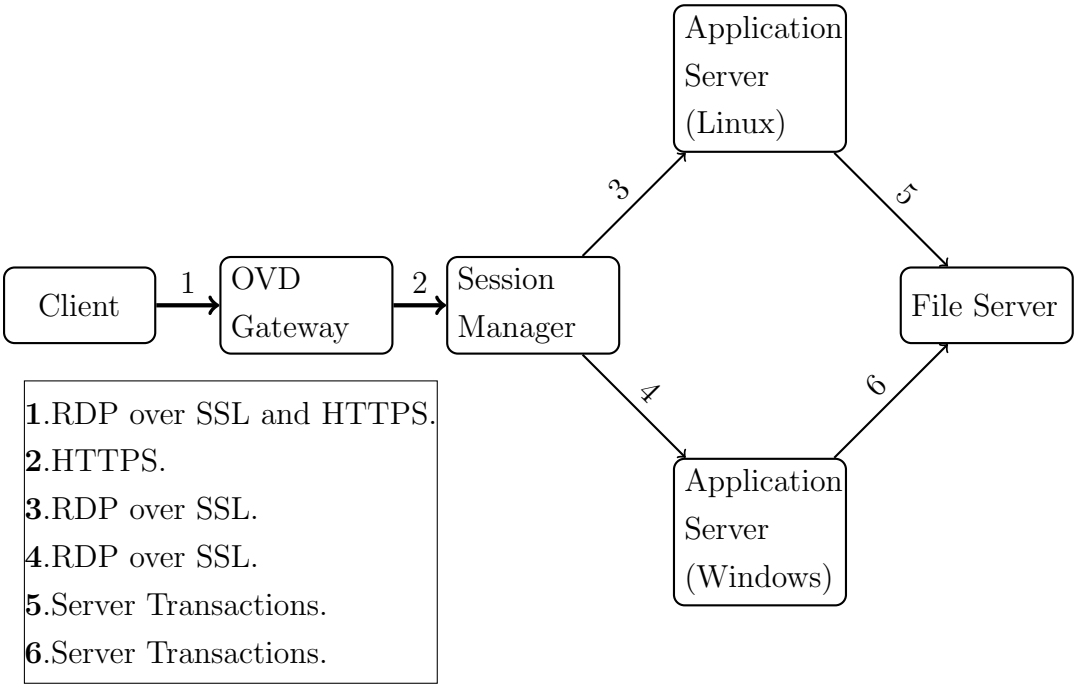

Figure 2.1: Architecture of OVD [7]

2.2.1 Software Components

There are four main components in OVD[7]. They are

1. Session Manager
2. Application Server
3. File Server
4. Gateway

We describe each of these components in the rest of this section.

2.2.1.1 Session Manager

Session Manager manages the administration of console. This component supports both Linux and Windows. Session Manager itself hosts a web server. This session manager contains all the users' account information (usernames and passwords). To store the users' account information, the session manager uses MYSQL database. The administrator of session manager is the creator of user accounts. The administrator also takes care of user privileges to access the applications. Whenever a client wants to connect to OVD, he/she sends a request to this web server. Then the session manager checks whether the user is valid or not. If valid, then the session manager mounts the respective user's account from the File server.

2.2.1.2 Application Server

This software must be installed in the system that contains the applications which will be accessed remotely. It supports both Linux and Windows applications. The main functionality of this server is to make applications available to different users with different privileges. When the installation completes, this application server is visible in the session manager and its IP address is displayed. Then the administrator configures these applications to give different privileges to different users.

2.2.1.3 File Server

This software has to be installed in the system that contains the filesystem which will be accessed remotely. It hosts both user and shared directories. It supports only Linux for now. It is possible to install both Application server and File server in a single system. This server is used to store the users' accounts – Files and Directories. When the user logs into his/her account, the user validation is done. Once the user is validated, the session manager mounts the user's files and directories from the file server in the session manager. From the session manager, through the web server, the user can access his/her files and directories using the browser.

2.2.1.4 Gateway

OVD uses several ports during a session: mainly HTTPS(443) and RDP(3389). Some sites may use firewalls to block the RDP port. In all cases, however, the HTTPS port is open in the firewalls. The purpose of Gateway is to tunnel all the

OVD connections into an HTTPS link, thus allowing access from anywhere to the OVD session.

2.2.2 How it Works

The user who wants to use the web desktop should have an account in the session manager and appropriate permissions to access the applications.

- Firstly, the user logs into the web server which is hosted along with the session manager by providing the authentication details.
- Session manager checks the authentication details with MYSQL database which already exists in the session manager.
- Once the validation is done at session manager side, session manager mounts the user files and directories from his/her respective file server into the web server of the session manager.
- Once the mount part is done, the user can access the files and directories through the web browser.
- Session manager also makes available the allocated applications to the user from his respective application server.
- To make the applications available to the users, the session manager uses Remote Desktop Protocol(RDP).
- When the user is accessing the applications, all the changes are saved into the mounted files of the user.
- When the user logs out from the web server, the files and directories are automatically unmounted from web server.
- If any application is running when the user logs out of the system, those applications are automatically killed.

2.2.3 A Simple Session Creation

A simple session creation is shown below without a gateway and a fileserver.

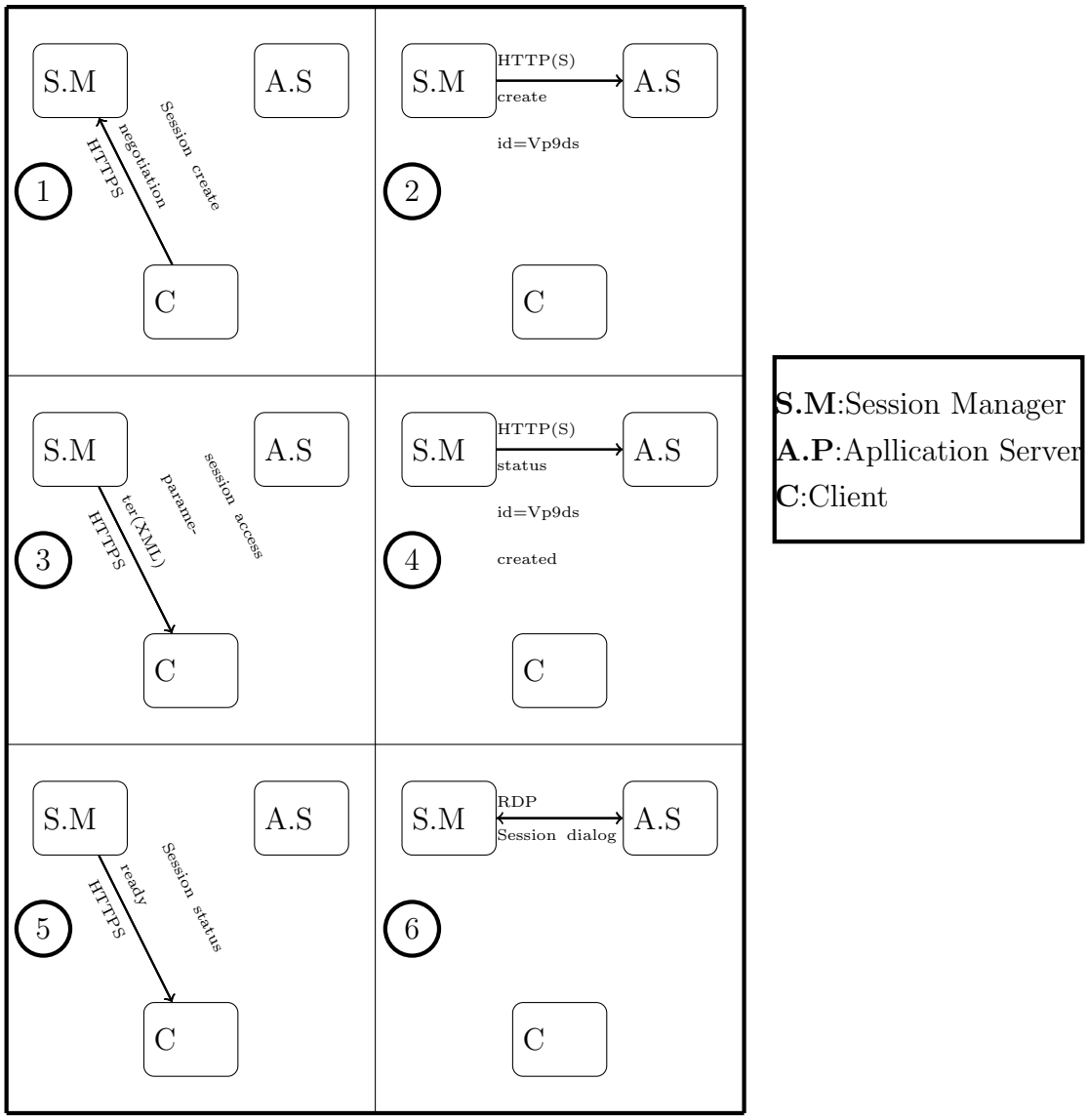

Figure 2.2: step by step session creation

2.2.4 Limitations

The following are the limitations of OVD:

1. OVD can only access the files and directories created in the session manager. Unless this is synchronized with the actual home directory of a user, access to the home directory is not possible. There is no feature to synchronize the session manager contents with the home directory of a user. The user needs to identify the files that are to be shared using the web desktop and host them on the session manager. Any modifications done on these files need to be synchronized manually with the user home directory.

2. Creation of user accounts in MySQL does not allow for many other features that are part of typical user authentication schemes such as LDAP to be used.

3. It requires the use of Remote Desktop Protocol (RDP) for access to applications. This is an additional protocol that needs to be installed on server side. RDP is also a heavyweight protocol that can hurt scalability.

2.3 Lively Kernel

2.3.1 What is Lively Kernel

Lively Kernel [2] is an open source web programming environment, developed by Sun Microsystems. It supports web development applications with nice graphics and ability to manipulate directly without any installation problems of normal desktop application. All the objects can be moved, resized, rotated and scaled freely. New objects can be composed by using existing objects by programming or visually. The entire system is written in javascript and it becomes active as soon as the page is loaded by the browser. This environment looks like a normal web page.

2.3.2 The Lively Dock: Web Applications as Live Thumbnails

[6] Lively Dock is an extension to Lively Kernel. Lively Dock is a user interface mechanism used to launch and minimize active web applications within the browser. This helps us to easily manage a large number of simultaneously running web applications in the web browser. Lively Dock is also written in javascript. The Lively Dock uses object transformation capabilities (mainly scaling) provided by the underlying Lively Kernel graphics framework.

2.3.2.1 Functionality

The Lively Kernel allows applications to be minimized and closed. However, this is applicable only to those that have a toolbar. Minimization results in showing only the toolbar and does not truly minimize it into an icon or a minimized thumbnail image.

The Lively Dock shows the minimized applications as live thumbnails. If all windows are minimized, all thumbnails of minimized applications are shown as an

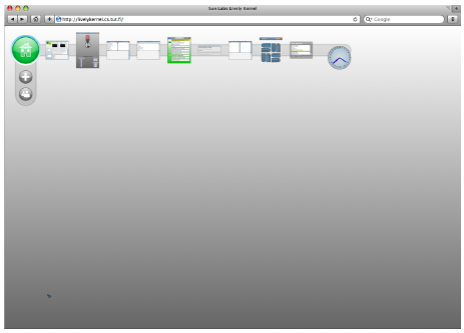

Figure 2.3: Lively Dock with thumbnails [6]

active horizontal taskbar. This taskbar is called as Lively Dock. When the mouse hovers over any of the applications, it shows the respective application name. The applications are still running even though the applications are minimized. The application objects (thumbnails) shown in the Lively Dock are not real thumbnails. The application objects are real application objects with significant smaller size. For this they have used object transformation provided by underlying graphics framework of Lively Kernel. This is shown in Fig. 2.3[6].

2.3.2.2 Additional features

- An additional feature is that Lively Dock itself can be minimized. The Lively Dock can be minimized in three states.

    1. The first is the normal state, i.e., a fully enabled state with all the applications and with some function buttons shown on Dock as shown in Fig. 2.4 [6].

Figure 2.4: Lively Dock with minimized applications

    2. In the second state, the Dock is displayed with function buttons only as shown in Fig. 2.5 [6].

Figure 2.5: Lively Dock with function buttons only

3. In the third state, the Dock is entirely minimized and displayed as a single thumbnail, shown in Fig. 2.6 [6].

Figure 2.6: Lively Dock with minimized dock

- The Lively Dock contains a number of function buttons. Plus(+) is one of the function buttons which is used to open a new application.
- The position of Lively Dock can also be changed. If the minimized Lively Dock is placed to the right on the screen and maximized, the horizontal bar expands to its left.

2.4 eyeOS

eyeOS is an open source web desktop. It is an operating system which gives us an environment like a normal desktop including standard applications such as file manager, contact manager, email reader, calendar, etc. within the web browser. The key benefits are privacy,file access over the Internet and the use of common applications irrespective of the OS used by the users. It also allows a user to download and upload files from and to the remote system.

Although it runs within the browser, it looks like a standard desktop with its menu, toolbar and user preferences. To access *eyeOS*, one should have an account on the remote *eyeOS* server. It is not built to access existing user home directories through the web browser. Once the user logs into the system, a full desktop session starts on the server as well as the client's web browser. To deploy the full desktop into browser, they use PHP application on server and browser uses Javascript and AJAX to run.

The application opened by the user will be launched as a different process in the remote server. The *eyeOS* desktop process communicates with the launched process in the backend server. *eyeOS* comes with some applications like office tools, networking, utilities etc.. To install new applications on the server, *eyeOS* needs administrative access.

2.4.1 System structure

The *eyeOS* system components are divided into two parts:

- The part run on client system (web browser).
- The part interpreted by the PHP server (web server).

The client side is used to render the user interface using Javascript and the server side uses PHP to carry out the operations.

2.4.1.1 Components

The *eyeOS* system structure contains four different components as shown in 2.7.

Kernel The core component that manages the implementation of the remaining components that form the *eyeOS* engine.

Services These are the components needed for *eyeOS* to work and manage its own internal operation. For example, a component that is responsible for managing users is a service.

Libraries The components responsible for providing support and security to *eyeOS*.

Frameworks These are non-essential components that add functionality to *eyeOS*. But these do not provide *eyeOS* operations on their own. An example is the office converter. It supports converts Microsoft office files and Open Office files and converts to pdf, eps, jpg, etc. as normal office converts.

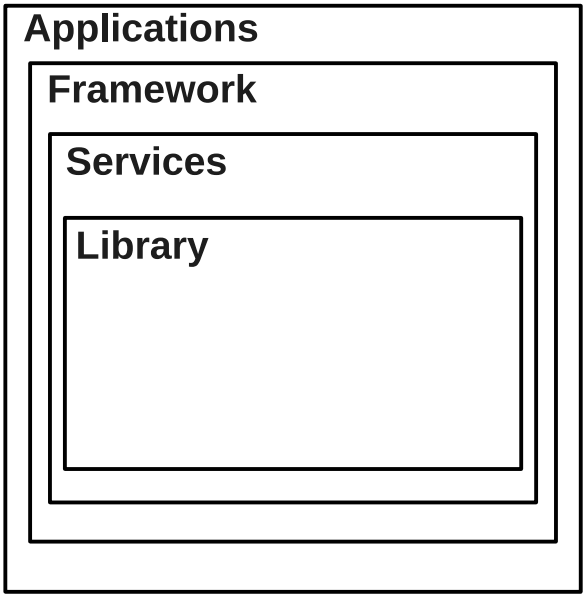

Figure 2.7: Components categories [1]

All components in *eyeOS* fit into these categories only.

2.4.2 Architecture view

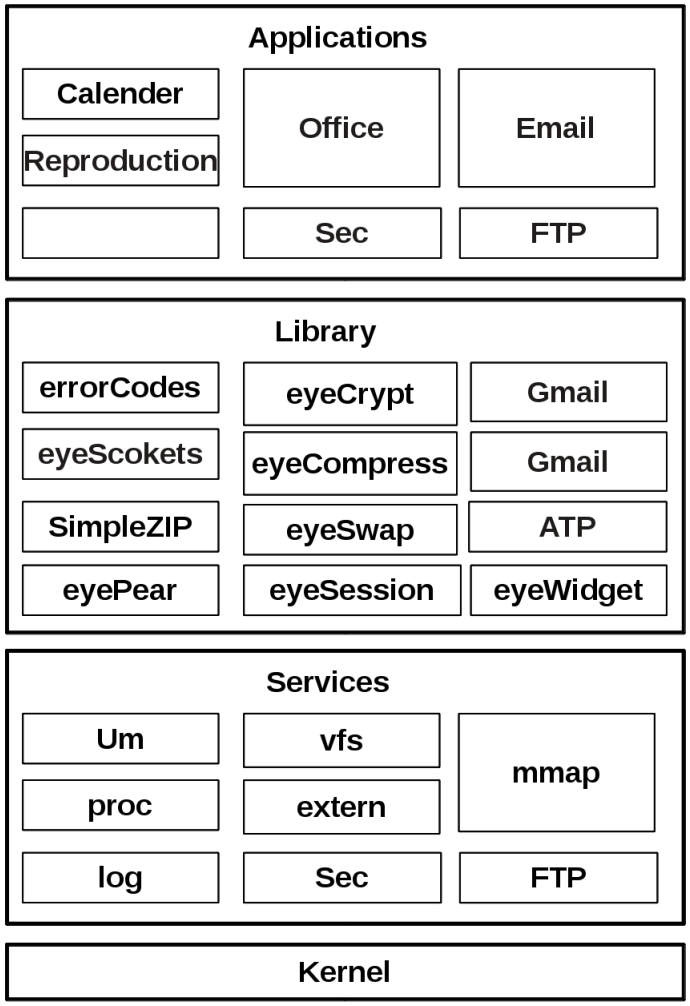

Figure 2.8: Architecture of eyeOS [1]

Service-MMAP This is a vital component in the communications of applications. It sends messages to client application. This is an automated service.

Service-VFS Its an *eyeOS* file system. It creates internally two files for each file.

file content

file information

Service-EyeX Gets the XML messages and responds in the same format to the browser as shown in Fig 2.9.

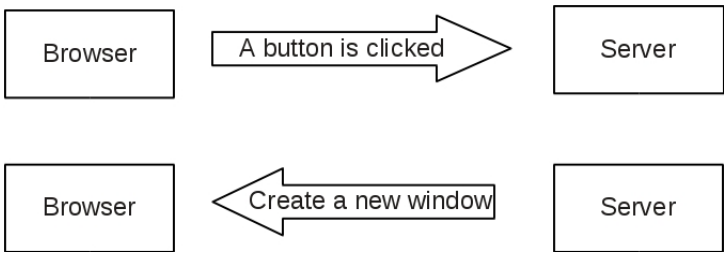

Figure 2.9: Sending messages [1]

Service-Proc It defines important variables for the application

mypid: to identify the each process uniquely.

checksum: to identify the process in client-server communication.

2.4.3 The Process inside eyeOS

Every time an application is executed, an entry in the process table of a user is created. This entry is an associative array that contains information about the name, Process ID (pid), checksum, tty, currentUser (the name of the user who launched the process) and the time it was launched. In the default installation, the process table is a session variable named 'ProcessTable'. By default an eyeOS application can have multiple instances running simultaneously. Every instance has its own process table and separate information. The administrator can configure an application to have only one instance. This is achieved by verifying that the process name is not found in the 'Process Table'.

2.4.4 Security

Some of its security measures involve the use of firewalls, backups every 24 hours and formatting a month as a preventive measure in case some are infiltrated by malicious code. As an additional precaution, the content stored by the users is encrypted. This prevents access of data even by the administrators.

2.5 Remote Desktop on thin clients for Virtual Machines [5]

Now-a-days the desktops or laptops have powerful hardware configuration. At the same time, due to advances in hardware, these powerful machines become obsolete

very soon. They need to be replaced with more powerful machines due to lack of service support and compatibility with newer operating systems. This results in increased costs to an organization.
In [5], the authors propose a low cost open source Remote Desktop to users to overcome this problem. The users can continue to use the existing thin or thick clients. However, unlike the typical case of thin clients connecting to a single remote server on which all the users are logged in and their processes run, the authors propose a remote desktop that allows load balancing and personalization of the operating system. They do this by using virtualization technology and existing open source software and tools. The tools used are:

- Linux Terminal Server Project(LTSP) for remote access.
- Xen hypervisor to provide virtual desktop environment.

The Remote Desktop [5] proposed allows each user to own a remote virtual machine which executes on a remote server and accessed by a thin client. Thus, the user will get a full fledged Remote Desktop with desired OS, instead of a simple session login on a single remote server. The live VM migration available with virtualization can help in load balancing and resource conservation. Complete user isolation can also be achieved by virtualization technology.

2.5.1 System details

The main open source components used are: Xen hypervisor to provide virtualization environment and LTSP to access the remote desktop. Some scripts are added to the LTSP software to map the user to a virtual machine (VM). This setup has 2 main components as shown in Fig 2.10:

- Management Server
- Virtualization Aware Physical Machine(VAPM)

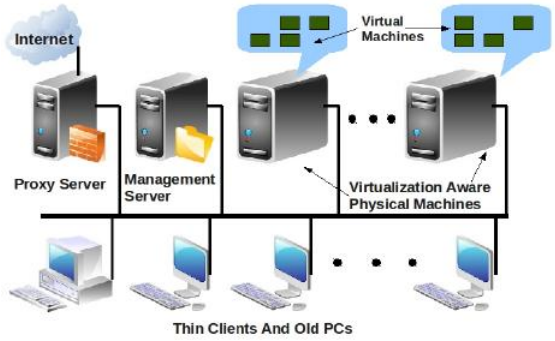

Figure 2.10: Solution approach for Remote Virtual Desktop access [5]

The Management Server is the central entity whose main task is to perform mapping between users and VMs. It also creates a VM on any one of the VAPMs and provides resources to users' VMs as per the user's requirement.
VAPM holds the users' virtual machines running on the installed Xen hypervisor. The user can access the VM without local hard drive and also can access the USB attached devices of the thin client.

Features of the Remote Desktop: The following are the features supported by the remote desktop proposed [5]:

User to VM Mapping: A user can use any thin client to access the remote desktop. So, the software maps the user to the VM using some scripts in LTSP.

Starting VM on User Log in: After a user logs in, his/her VM is started and the VM state is stored when the user logs out.

Provisioning of VM to Host Virtual Machine: Based on previous history, the user's approximate resource requirements are estimated and reserved.

Enhancing System Efficiency: Dynamic load balancing and consolidation provided by Xen are used to enhance system efficiency.

Chapter 3

SNAPWebD: Proposed Web Desktop

The products which we discussed in the previous chapter are not having the capability of accessing the user account in an NFS Server through the web browser. Most of the existing products are commercial. The installation of these softwares is complex or cumbersome. Some of them require additional software to be installed at the client side. Further, they use some additional protocols to configure the web desktop. We have proposed a web desktop, **SNAPWebD**, that overcomes most of these limitations.

3.1 Architecture of SNAPWebD

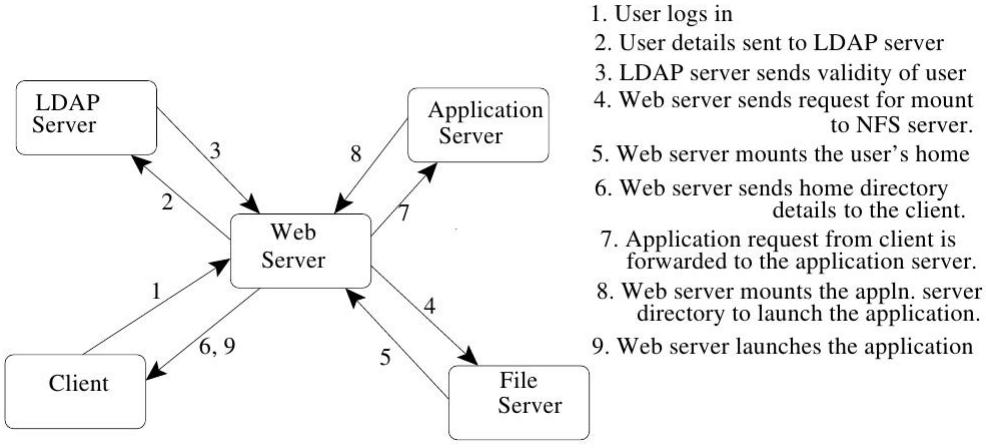

Figure 3.1: **SNAPWebD** Architecture

The proposed **SNAPWebD** architecture consists of the following components and is shown in Fig. 3.1:

1. Lightweight Directory Access Protocol (LDAP) Server
2. Network File System (NFS) Server
3. Web Server
4. Client
5. Application Server

The web server is the only system where **SNAPWebD** software has to be installed. The **SNAPWebD** software is a lightweight software, which consist of 2000 lines of JSP and JQuery code. So, installation and maintenance of this product is easy and simple. In the rest of this section we describe the various components and their purpose.

3.1.1 LDAP Server

We have used OpenLDAP 2.4.35 [4] software which is free and developed by OpenLDAP Project.

3.1.1.1 OpenLDAP features

- LDAP is a lightweight alternative to the X.500 Directory Access Protocol.
- LDAP is a platform independent protocol.
- LDAP also has other simplifications such as representing most attribute values and many protocol items as textual strings, that are designed to make clients easier to implement.
- We are using OpenLDAP to provide centralized authentication for users to login from anywhere in the network.
- LDAP is not in any way related to traditional system usernames or other data. However, part of its functionality of our setup will consist in storing information traditionally found in Linux files /etc/passwd and /etc/group, thus making that data network-accessible at a centralized location.

3.1.1.2 How LDAP Works

The OpenLDAP server (*slapd*, the Standalone LDAP Daemon) architecture was split between a frontend which handles network access and protocol processing, and a backend which deals strictly with data storage. The architecture is modular and many different backends are available for interfacing to other technologies, not just traditional databases. Whenever a user is registered (entry created in database) in the LDAP server, it stores the information in a tree structure. A user directory (entry of a user) contains a lot of attributes such as name, password, mail, uid, givenName, homeDirectory etc.. If we request for an entry, i.e., a user directory information, in the LDAP server, we need to give full path, from the top node to the user node, of the tree. LDAP server receives this information through its frontend and passes it to the backend for processing. When the backend completes a request, it returns the result to the frontend, which then sends the result, i.e., the whole directory structure, to the requested client.

3.1.2 Web Server

The web server we have used is Apache Tomcat. Apache Tomcat is a servlet container developed by Apache Software Foundation(ASF). Tomcat implements the Java Servlet and the Java Server Pages(JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. There are different versions of Tomcat available. We have used the version Tomcat 7.0.30. Its features are:

- It implements the Servlet 2.4 and JSP 2.0 specifications.
- It has reduced time for garbage collection and improved performance and scalability.
- Faster JSP parsing.

3.1.3 NFS Server

Network File System(NFS) is a distributed file system protocol developed by Sun Microsystems. It allows a user on a client computer to access files over a network as if they are local files. Another advantage is also that the user is not required to know whether the accessed files are local or remote. In our project, we are using NFS version 4.1. [3].

3.1.3.1 How NFS Works

- The server has an NFS daemon, $nfsd$, in order to make the data, both files and directories, available to clients.
- The server administrator determines what to make available, exporting the names and parameters of directories by using the '/etc/exports' configuration file and 'exportfs' command.
- The server can also determine valid clients, so that only those clients can access the exported directories.
- The firewall should open a port for $nfsd$ daemon.
- The client requests the exported data by using 'mount' command.
- If all goes well the user can access the mounted filesystem with given permissions – read, write and read and write.

NFS mounting are of three types: one is *bootup mount* where all exported data are mounted into the client system at bootup time itself. Bootup mount can be done using '/etc/fstab' configuration file. Second type is to mount these manually using the *mount* command by a user. A third type of mount is *automount* where exported directories are mounted, whenever they are required without the intervention and knowledge of the user. This is useful in reducing network traffic as well as adding additional application NFS servers without too much configuration. Automount can be done using '/etc/auto.master' configuration file..
We used NFS automount for **SNAPWebD**.

3.1.4 Client

The client can be any system from where the user will access his/her own home account through the web browser. It can be a thin client or a thick client such as standard desktop or more portable devices such as laptop, tablet, smart phones etc..

3.1.5 Application server

This is the place where the all the applications are hosted. Through this server the applications are accessed transparently through NFS exports.

The application server is not necessary in the **SNAPWebD** product. If an organization wants to use an application server which can host all applications in it, **SNAPWebD** also supports this feature.

3.2 How SNAPWebD Works

The **SNAPWebD** works as follows:

1. Firstly, the login page is displayed when a user types the URL for the **SNAPWebD**.

2. Then the user details are accepted by the software in the web server and these are sent to the LDAP server to authenticate the user.

3. The LDAP server returns an exception, if the user is not valid. The web server returns an error message to the user.

4. If the user is valid, the LDAP server returns the home directory information of the user who has logged in.

5. Using the home directory information, the web server software checks if the NFS logical volume corresponding to the user's home directory is already mounted. If it is not, it mounts the NFS logical volume automatically using the 'auto mount' option from the respective NFS Server.

6. Then the home directory contents are sent to the web browser which displays them.

7. If the user wants to launch an application, the **SNAPWebD** software searches in the local system for the application.

8. If the required application is not found in the local system, then the web desktop software searches in the web server, NFS Server and Application Server if it exists respectively. This is done in this order to increase the performance of the system.

9. If we click any file in the browser, the web server software launches the corresponding application and opens the clicked file with the launched application. If an application that corresponds to this file is not found in any of the servers, it returns an error to the user.

10. When the user logs out of the system, all the launched applications will be killed and the home directory of user is unmounted.

The launch of an application on the remote server takes place by using *ssh* protocol. Using this *ssh*, the **SNAPWebD** software launches an application by starting a process on remote server and exports the display to the system the user has logged from.
To launch the application the search sequence – local system, web server, NFS server, application server – plays a vital role in performance. As long as an application is available in the local system there is no need to connect to the remote server over the Internet to run applications remotely. This gives improvement in performance in three ways:

1. No network load.
2. No load on remote server in terms of running more applications at remote side.
3. Working with local process is faster than working with a remote process.

3.3 Features of SNAPWebD

In this section, we discuss in detail the various features offered by the **SNAPWebD**.

3.3.1 User with His Personal Desktop

with asking user authentication details as shown in Fig. 3.2. The first step of using **SNAPWebD** is to go to the URL specified for it. This displays the login page. Once a user logs into the system by giving the authentication details, the user can view his/her home directory on web browser. The pages with user home directory will be shown as illustrated in Fig. 3.3.

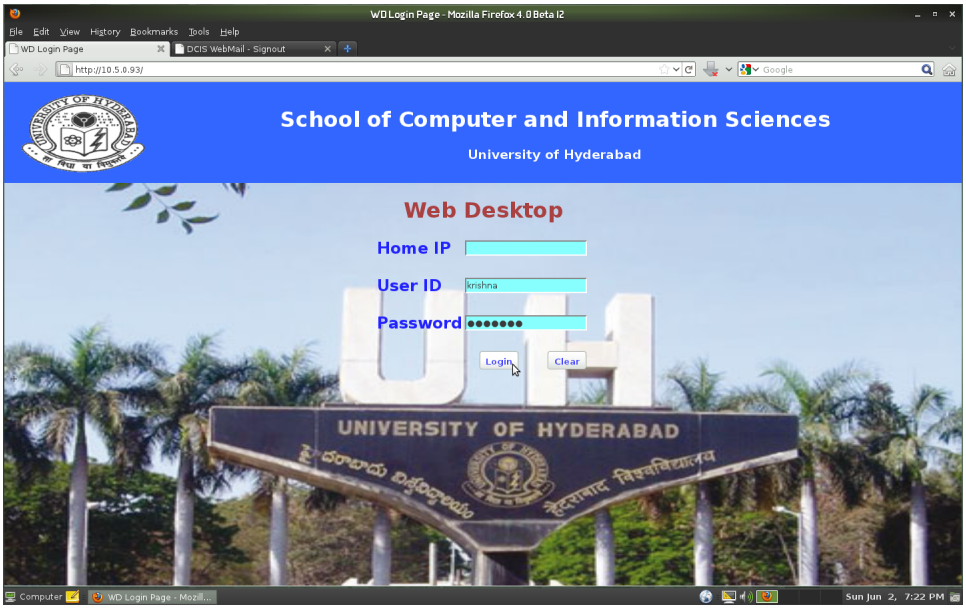

Figure 3.2: Login Page

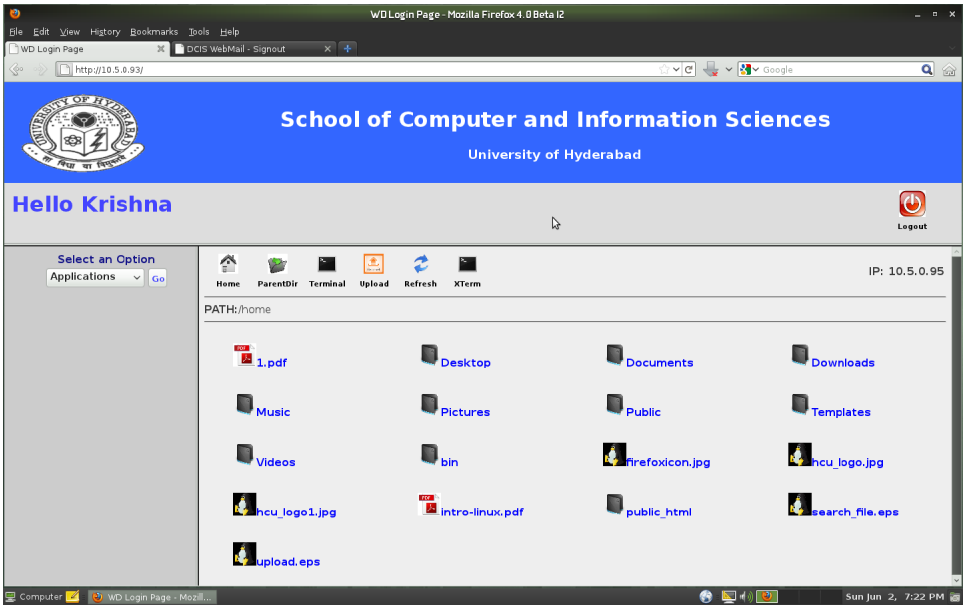

Figure 3.3: User Home Page with His Home Directory

3.3.2 Opening a File

Once a user logs into the **SNAPWebD**, the user is able to common phrases students use. See the correction made. see his/her personal desktop in the browser. If the user wants to open a file, a click on that file will take the action associated with the type of the file as specified in the plugins of the browser. This means that when the user clicks a file, the **SNAPWebD** launches an application by seeing the

type of file and opens the file with the launched application for all file types configured. This is shown in Fig. 3.4.

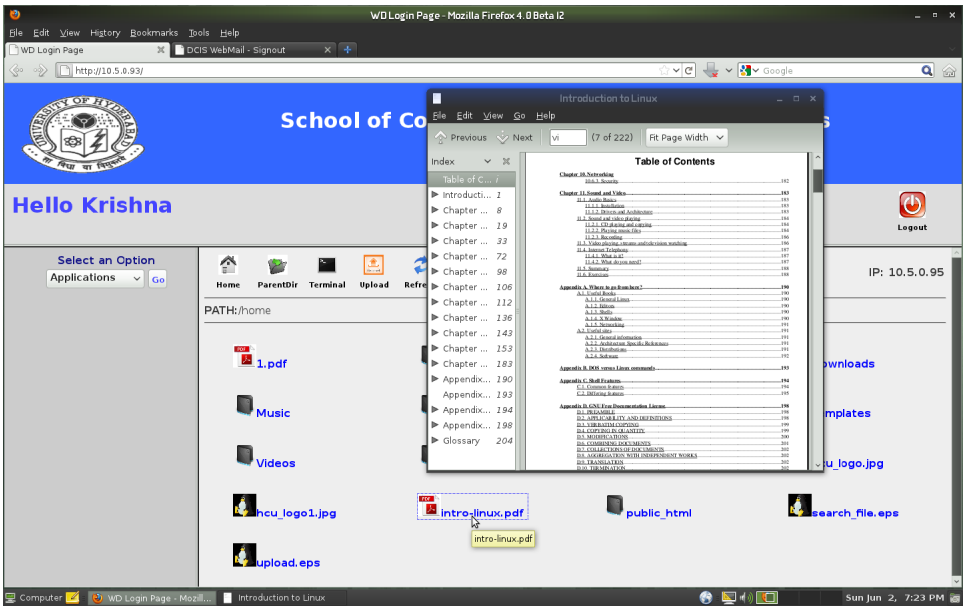

Figure 3.4: Opening a File

3.3.3 Upload and Download the Files

Users can upload any file from the local system to remote NFS Server where the user home accounts actually exist and download any file from his/her home account to the local system. The right click on any file will have options like “Download” and “Open With” etc.. Using the “Download” option, the user can download the files from remote system to local system as shown in Fig.3.5 and Fig.3.6.

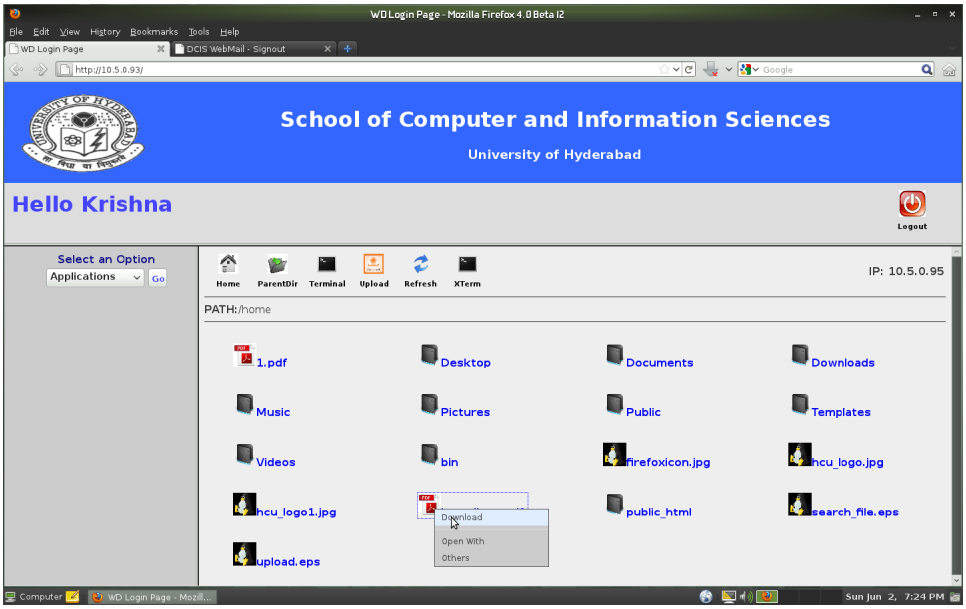

Figure 3.5: (a)Downloading a File

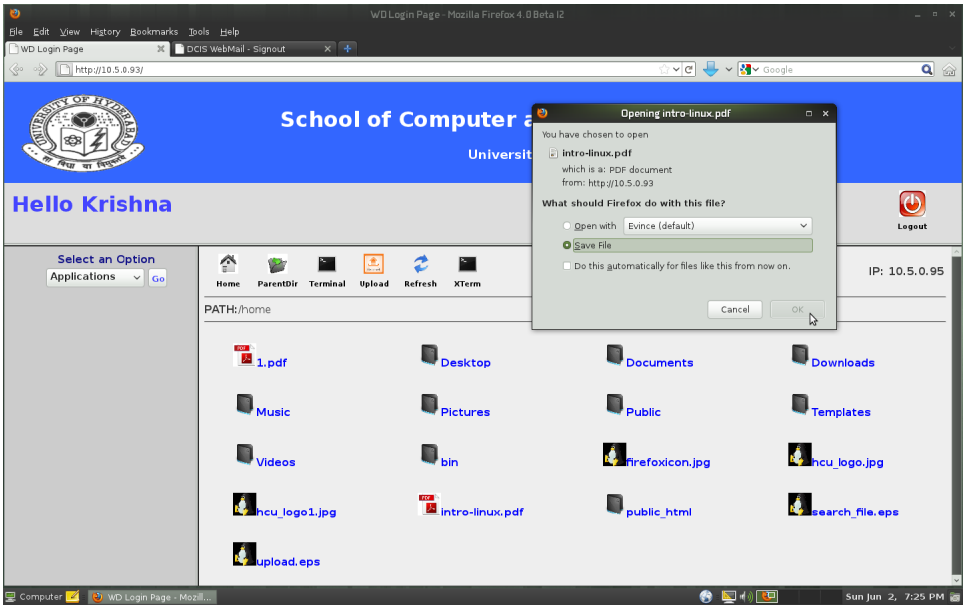

Figure 3.6: (b)Downloading a File

If a user wants to upload a file from the local system, he can do it with a button click named as upload which is at the top of the page. The uploaded file will be stored in currently viewed directory. This is shown as Fig. 3.7.

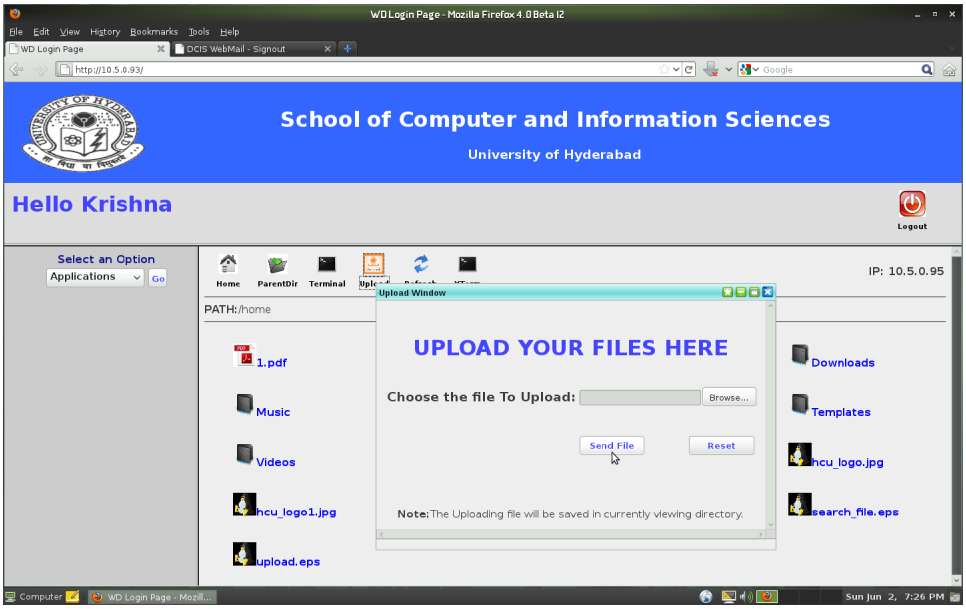

Figure 3.7: Uploading a File

3.3.4 Opening The Terminal

SNAPWebD provides two type of terminals:

- Lightweight Terminal
- Complete Terminal.

3.3.4.1 Lightweight Terminal

To open the terminal, click on *Terminal* button at the top of the page. This is shown in Fig. 3.8. Because the software is written in Java Server Pages (JSP), it is a lightweight process. When the user types the command in the lightweight terminal, the **SNAPWebD** connects to the Remote NFS home directory's server using *ssh* and executes the command there and gets the result to the client. When the user gives a command, then only it connects to the Remote NFS Server; otherwise, it will be idle. Therefore, the load on the network is less. But here we cannot execute interactive commands in lightweight terminal. To do those we need to go for a complete terminal such as an *xterm*.

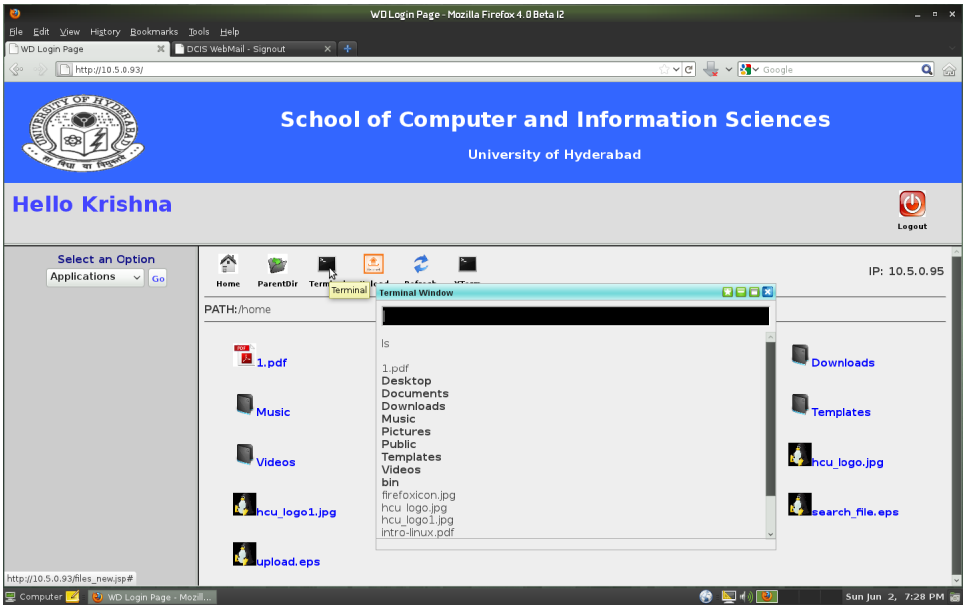

Figure 3.8: Lightweight Terminal

3.3.4.2 Complete Terminal(XTerm)

A complete terminal is creating a process by launching an *xterm* application on Remote NFS File Server using *ssh* and exporting its display to the client system. To open *xterm*, click on *XTerm* button on the top of the page. This is shown in Fig. 3.9.

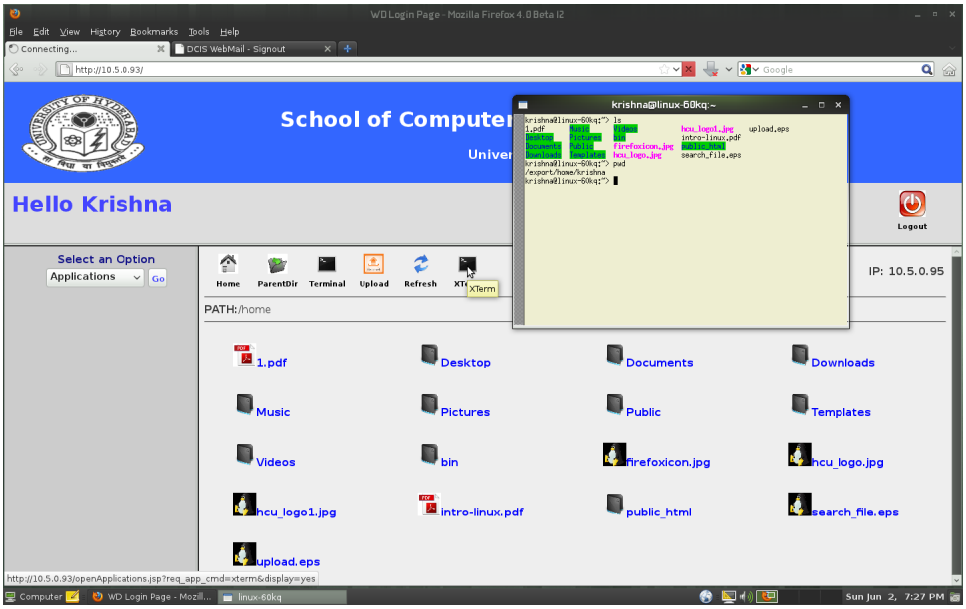

Figure 3.9: A Complete Terminal(XTerm)

3.3.5 Search

SNAPWebD provides 2 different type of searches:

1. Application Search
2. File Search
   (a) Name based search
   (b) Content based search

3.3.5.1 Application Search

When we select 'Application' option in drop-down box menu of left frame of the page, it displays a text field which asks for an application command to search. If we type a command in that, the **SNAPWebD** will try to launch the application in the Remote NFS file server, if it exists. If there is no such application in that system, it returns an error message. This is shown in Fig. 3.10.

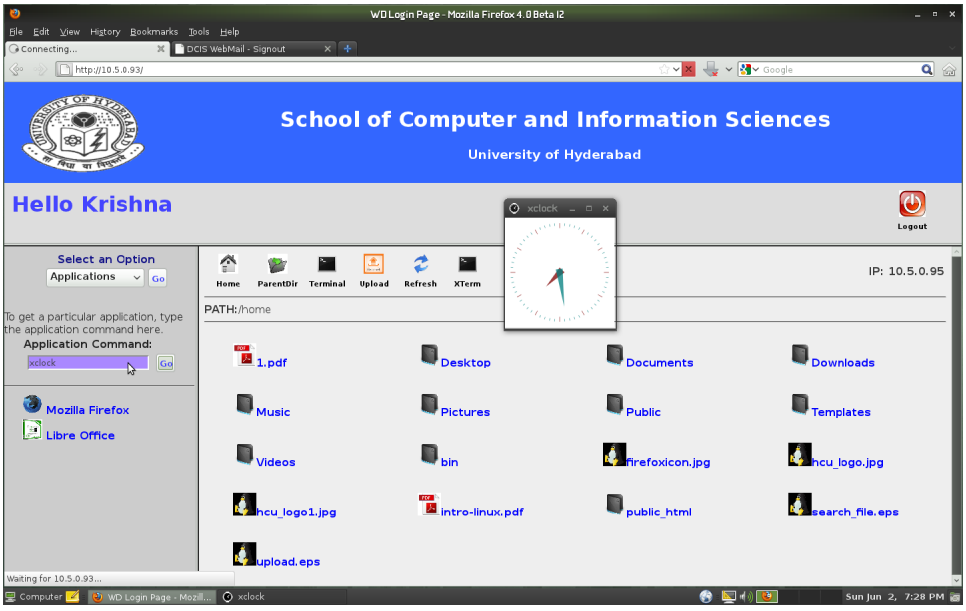

Figure 3.10: Search for "xclock" application

3.3.5.2 File Search

When we select 'search' option in drop-down box menu of left frame of the page, it provides name based search and content based search.

Name based search: In name based search, we can search for files based on the name of the file using wildcard character as is true for typical command line search. All files and directories that match the given regular expression will be displayed in the left frame of the page. This is a recursive search on the entire home directory of the user. This is shown in Fig. 3.11.

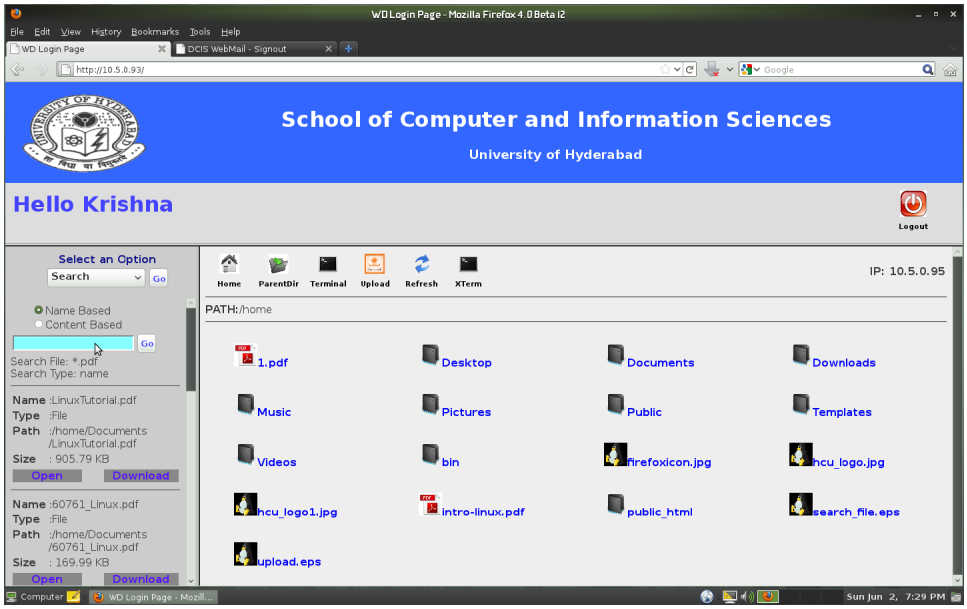

Figure 3.11: Searching for all pdf files

Content based search This is a rudimentary search on the content of the files. The given string is searched for in the files of the home directory recursively and all the files that contain the given string are displayed in the left frame of the page. This is not an accurate search. In future, we plan to expand this feature to have a more sophisticated content based search. An example content based search is shown in Fig.3.12.

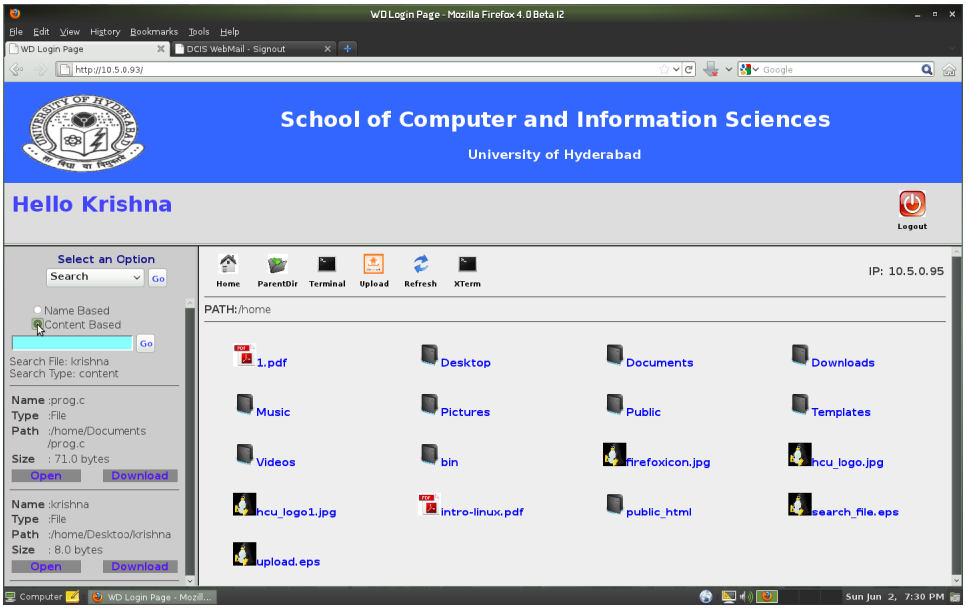

Figure 3.12: Search for files containing “krishna” as text

3.3.6 History

The **SNAPWebD** maintains the history of all the actions done by the user since from the login time. The actions include opening a lightweight terminal, all the commands executed in the lightweight terminal, launching an application, searches for any file or directory in the whole system, uploading and downloading any files from server etc.. When we select the ‘history’ option in the drop-down menu of the left frame of the page, all the history will be displayed in that left frame. This is shown in Fig. 3.13.

Figure 3.13: History of Actions

3.3.7 Accessing the Specified Remote Computer

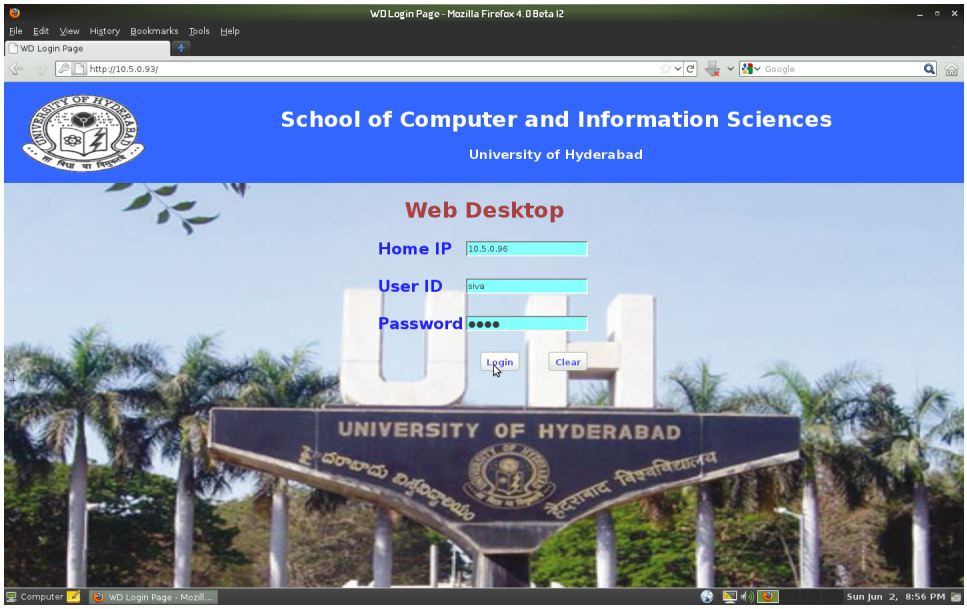

Figure 3.14: Login Page

An additional feature of the proposed **SNAPWebD** is that it allows access to a local user account on a given system identified by its IP address or a logical name. This is shown in Fig. 3.14. We assume that this system runs the NFS server software. In this case, the **SNAPWebD** software authenticates the user locally without using the LDAP server. Once the user is authenticated, the home directory

of the user on the remote system is mounted on the web server and the user can access the files and directories of the specified remote system. The history of commands executed on the specified standalone system is shown in Fig. 3.15. assuming from the name of the label "history-ip" that you are showing the history of commands on a system with the given IP.

A user can access an NFS account and a local account on a different standalone system simultaneously using the **SNAPWebD**. Files can be transferred from one system to the other by downloading from one system to the local system and then uploading it to the other. In future, we plan to develop a feature to drag and drop files from one web desktop to another web desktop.

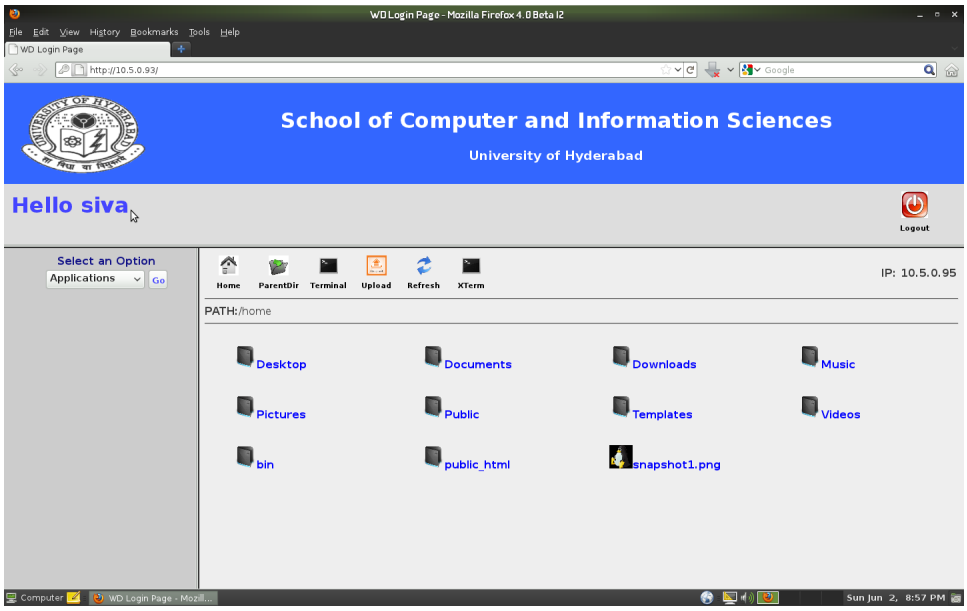

Figure 3.15: User Home Page with His Home Directory

3.4 Summary of Features

A table comparing the features supported by various commercial and free software web desktop products and **SNAPWebD** is given below.

Table 3.1: Comparison of Features supported by **SNAPWebD** and Other Products

Features	**Products**			
	VNC	**eyeOS**	**OVD**	**SNAPWebD**
Display files and directories in a browser	NO	YES	YES	**YES**
Can a user get his personal desktop	YES	NO	NO	**YES**
Opening a file when the associated application does not exist in the remote system hosting the file	NO	NO	NO	**YES**
Ability to upload a file to the remote host from local system	NO	YES	YES	**YES**
Ability to download a file to the local host from the remote system	NO	YES	YES	**YES**
Lightweight terminal (execute all types of non-interactive commands)	NO	NO	NO	**YES**
Complete terminal (execute all types of commands)	YES	NO	NO	**YES**
Search for an application in user's remote system	YES	NO	NO	**YES**
Search for a file based on name	NO	NO	YES	**YES**
Search for a file based on content	NO	NO	NO	**YES**
Store the history of users' actions since login time	NO	NO	NO	**YES**
Accessing a specified standalone system (using its IP address)	NO	NO	NO	**YES**

Chapter 4

Conclusions and Future Work

We have developed a web desktop, **SNAPWebD**, that allows users to access their home directories hosted on NFS servers using the browser. While there are other web desktop applications available both free and commercial products none of these work with existing infrastructure of most institutions/corporations where users accounts are hosted on NFS servers. Our **SNAPWebD** is a way for users to access their home directories on any NFS mounted logical volume. Further, we do not need any special software to be installed on the client system in our product. Thus, this can be readily used by any user from any platform such as laptop, desktop, smart phones and tablets to access their home accounts without any modification to these systems. This helps in the ease of deployment. Our SNAPWebD is designed to search various file and application NFS servers hosted by institutions normally to launch the applications needed. To improve performance, the **SNAPWebD** uses all applications from the local system if they are available and from remote servers only if they are not. The **SNAPWebD** can be used to also work with command line by using the standard *xterm* or the lightweight terminal built by us as part of the **SNAPWebD**. Unlike the *xterm*, our lightweight terminal does not need to transmit every key typed to the server and back to echo it nor send the mouse clicks etc. over the network. This reduces the burden on the network and makes our terminal faster in response. The users can also download and upload files to and from the remote file servers.

One additional feature our **SNAPWebD** has is the ability to access any standalone remote system given its name or IP address in the login page. In this case, the user's account is a local account and not authenticated by the LDAP server. The files and directories of this standalone system can be accessed through the web browser once the user is authenticated.

4.1 Future Work

1. Currently the **SNAPWebD** is supported by linux systems only. In future it can be extended to support all operating systems.

2. The **SNAPWebD** can be extended to connect to Virtual Machines(VMs) and allow the users to access their VMs using the web browser.

3. Security issues related to this product need to be explored and fixed.

4. The feature to upload and download the files will be enhanced to a drag-and-drop operation.

Bibliography

[1] **EyeOS**. `http://www.eyeos.com/`.

[2] **Lively Kernel**. `http://www.lively-kernel.org/`.

[3] **Network File System(NFS)**.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.473`,
`http://tools.ietf.org/html/rfc5661`.

[4] **Open LDAP**. `http://www.openldap.org/`.

[5] Dhaval Manvar, Mayank Mishra, Anirudha Sahoo. **Low Cost Computing Using Virtualization For Remote Desktop**. In *Fourth International Conference on Communication Systems and Networks, COMSNETS 2012, Bangalore, India.*, 2012.

[6] Jan Miettinen, Tommi Mikkonen and Antero Taivalsaari. **The Lively Dock: Web Applications as Live Thumbnails**. In *International Symposium on Web Systems Evolution, WSE-2010, Timisoara, Romania*, 2010.

[7] Ulteo Organization. **Open Virtual Desktop(OVD)**.
`http://www.ulteo.com`, 2012. Technical Documentation.

Appendix A

User Manual

A.1 Login

To login into the **SNAPWebD** enter your username and password of LDAP server to view the NFS Server home account. To access your personal computer, enter your username, password and IP address of your system.

- If the authentication is done, you will be redirected to a page which contains all your file and directories of your home account.
- If your password is wrong, you will get an error message that your password is wrong.
- If the username is wrong, you will redirected to a page which shows that you are not a valid user.

A.2 Features

A.2.1 File Exploration

After logging into the **SNAPWebD**, you can view your home account contents (files and directories). If you want to open a directory, click on that directory or you have right click option "Explore". To go back previous page you can use normal browser back button. If you want to go parent directory you have a button *ParentDir* on the top of the page. There is also a button *home* on top of the page to get you directly to the home directory irrespective of your current path.

A.2.2 Upload and Download

To upload files there is a button *upload* at the top of the page.

- Click *upload* button to open the upload window.
- The window contains **Browse** button and **Send File** button.
- On clicking on the **Browse** button you can select a file you want to upload to your remote home account.
- Click **Send File** button to upload the file.
- On success, it will display a "File Uploaded Successfully" message; otherwise, no message is displayed.
- The uploaded file will be saved in the directory being viewed currently.

To Download any file from the server, we have a right click option "Download" on files. If you click that option the file will be downloaded into your local system.

A.2.3 Lightweight Terminal

To open a lightweight terminal, you have a button *Terminal* at the top of the page.

- Click on *Terminal* button to open the lightweight terminal. A text field is opened in that window.
- Type any Linux command in the text field and press Enter button.
- The results of that command are displayed in the lower part of the lightweight terminal.
- To close the terminal there is a cross(X) button on top right corner. Click it to close the terminal.

With this terminal, we cannot use interactive commands. To use such type of commands use *XTerm*.

A.2.4 XTerm

To open a terminal with full functionality, you have a button *XTerm* in top of the page.

- This is exactly like a terminal in your remote home system.
- Click on *XTerm* button to open the terminal.
- It will work for all type of commands.

A.2.5 Search

SNAPWebD provides you different types of searches:

- Application Search
- File Search

  Name based search

  Content based search

A.2.5.1 Application Search

In the left frame of the page there is a combo box with a heading "select an option". Select Application option and click on *Go* button. You will have the left frame with a text field with a *Go* button.

- Type your required application command on that text field and click on the *Go* button.
- It opens the application if it exists in your remote home account system.
- If not, it shows an error message to you.

A.2.5.2 File search

To find any file in your entire home directory, select "search" option on left frame of page and click *Go* button. You will find a text field with two radio buttons "Name Based" and "Content Based".

Name Based

- Select "Name Based" radio button.
- Type your required file name or a regular expression with wild card character(*) on that text field.
- Click *Go* button.
- It lists all the files with that name or regular expression.
- If not, it displays an error message to you.

Content Based

- Select "Content Based" radio button.
- Type your required content in the text field.
- Click *Go* button.
- It lists all the files with that content.
- If not, it shows an error message to you.

This search is not an accurate search.

A.2.6 History

History of actions is the set of actions done from login time to the present which can be seen in the left frame of the page when you select "History" button from combo box options.
The actions are like

- Uploading and downloading files.
- Search for a file or an application.
- Opening lightweight terminal.
- History of commands executed in lightweight terminal etc..

Appendix B

Installation Manual

To use the *SNAPWebD* product, we need to install and configure 4 components

- Web Server
- LDAP Server
- NFS Server
- Application Server

B.1 Prerequisites

B.1.1 System Requirements

- For a Web Server
  - Minimum prerequisites are 1 CPU core and 512MB RAM.
  - Recommends a minimum of 4GB RAM and 2 CPU cores.
- For an LDAP Server
  - Minimum prerequisites are 1 CPU core and 512MB RAM.
  - Recommends a minimum of 1GB RAM.
- For an NFS Server
  - Minimum prerequisites are 1 CPU core and 512MB RAM.
  - Recommends a minimum of 2GB RAM and 2 CPU cores.

- For an Application Server
  - Minimum prerequisites are 1 CPU core and 512MB RAM.
  - Recommends a minimum of 4GB RAM and 2 CPU cores.
- For a Client System
  - Any thin client which can have a web browser.

B.2 Web Server Installation and Configuration

The web server we are installing is Apache Tomcat which supports servlets and JSP. This is different from normal Apache web server which can deploy only HTTP pages.

B.2.1 Apache Tomcat Server Installation

- To install Apache Tomcat the prerequisites are
  - java
  - JDK
- Download Tomcat source package and uncompress it.

  tar xvfz apache-tomcat-7.0.30.tar.gz
- Copy the apache-tomcat-7.0.30 directory into a desirable directory and this path will become TOMCAT_HOME directory.(Let us take that TOMCAT_HOME directory is **/opt/apache-tomcat-7.0.30**)
- Setup the paths for catalina and others.

  CATALINA_HOME=/opt/apache-tomcat-7.0.30

  CATALINA_BASE=/opt/apache-tomcat-7.0.30
- Setup the JAVA and JDK paths to compile and run the JSP code (Let us assume that java 1.6 already exists in the system)

  JAVA_HOME=/usr/lib/jvm/jre-1.6.0-openjdk

  JDK_HOMe=/usr/lib/jvm/jre-1.6.0-openjdk
- Now start tomcat server.

  sh /opt/apache-tomcat-7.0.30/bin/startup.sh

- By default Apache Tomcat will be hosted on "localhost : 8080" port. So try to access the site with web browser.
- If you are redirected to apache tomcat server starting page, installation is successful. Otherwise check the 4 environment variables which are configured earlier are configured correctly or not.

B.2.2 Apache Tomcat Server Configuration

The two most important configuration files to get Tomcat up and running are called server.xml and web.xml. By default, these files are located at TOMCAT_HOME/conf/server.xml and TOMCAT-HOME/conf/web.xml, respectively.

- Edit the **/opt/apache-tomcat-7.0.30/conf/server.xml** file in apache tomcat server to make 8080 port to default(80) port.

  Connector port=8080 to **Connector port=80**

- Edit the **/opt/apache-tomcat-7.0.30/conf/web.xml** to
  - Specify default welcome page(e.g. index.jsp).
  - Change the default time for session expiry (default time is 30 min.)
- Deploy the *SNAPWebD* software into this Apache Tomcat server. This means copy the software into **/opt/apache-tomcat-7.0.30/webapps/ROOT/** location in Apache Tomcat Server.

To run the application remotely and export that application display to user, we use *ssh* protocol. The *SNAPWebD* software uses *sshpass* to avoid interactive *ssh* session. This sshpass package is available in `http://sourceforge.net/projects/sshpass/files/sshpass/`. Download the compatible version of the WebServer Operating System(OS) and install it. To know whether the *sshpass* package installed correctly or not, use the command:
sshpass -V

B.2.2.1 After NFS Server Installation

This configuration should be done after NFS server installation which is described in section B.4. The web server itself is an NFS client. After the installation of NFS

server, i.e., after the user's home directories are exported, we need to mount those directories in the web server (which is the NFS client). To mount the files from NFS server, we have 2 ways

1. **Boot up method** The exported directories are mounted from NFS server at boot time it self. For this we need to configure */etc/fstab* file
2. **Auto mount method** The exported directories are mounted from NFS server when ever the directories are required. For this we need to configure */etc/auto.master* file.

We are using **Auto mount method** in this project. So, to enable auto mount method we need to start the *autofs* deamon by using command: **service autofs start**.

The configuration of */etc/auto.master* file is as follows:

- Write the configuration line into */etc/auto.master* as

  TOMCAT_HOME/webapps/ROOT/any-new-directory/ any-file-containing-the-mount-configuration

  e.g. /opt/apache-tomcat-7.0.30/webapps/ROOT/IMPORT/ auto.misc

- The mount-configuration-file contains the configuration line as

  any-new-directory options NFSserver-IPaddress:NFSserver-exported-directories(user's home directories)

  e.g. home -rw 10.5.0.95:/export/home

  - home: A new directory
  - -rw: read and write permissions
  - 10.5.0.95: NFS Server IP address
  - /export/home: NFS server exported directory.

B.2.2.2 After Application Server Installation

If the application server does not exist, no need to do this configuration. If application server exist, This configuration should be done after application server installation and configuration which described in section B.5.
Mount the exported */usr* directory into web server by using **Auto mount method** option.

B.3 LDAP Server Installation and Configuration

An open source software called *OpenLDAP* software is used to install LDAP Server. There are also many other LDAP Server softwares which are commercial.

B.3.1 OpenLDAP Installation

We can do the installation in two ways - using GUI or the command line.

Using GUI

- Go to Software Management in Yast2.
- Search for "yast2-ldap-server".
- Install the software with root privileges.
- After the installation, check for "LDAP Server" icon in "Yast2". If it exists, installation was successful.

Using command line

- Download the OpenLDAP software from openldap official site `http://www.openldap.org/`.
- Copy the package to a desirable directory and uncompress it.

  tar xvzf openldap-2.2.5.tgz

- Type **./configure** command to build the software with default options. For customized installation type **./configure –help** command to know the options. Watch the output to see if all went well.

- After configuring the software you can start building it. First build the dependencies, using the command: **make depend**.
- Build the server after that using the command: "**make**".
- To ensure the correct build, we should run the test suite with the command:**make test**.
- Now install the binaries and man pages with supervisor privileges. **su root -c 'make install'**.

B.3.2 OpenLDAP Configuration

Once the OpenLDAP software has been built and installed, we are ready to configure it. The configuration has the following steps:

- Go to Yast2 and click on LDAP server icon to open the LDAP server.
- Start the LDAP server and open a port (the default port for LDAP is 389) in firewall to access the LDAP entries by the web server as shown in Fig.B.2
- Then enter the Basic DN (Basic Domain name), administrator common name and passwords as shown in Fig.B.3
- If all goes well, you will get a message that the server has started. This is shown in Fig.B.4

B.3.2.1 Adding Users to LDAP Server

Create user.ldif and add it to the LDAP server with *ldapadd* command.

- The example snapshot of user.ldif file is given below

```
dn: uid=sai,ou=webdesktop,dc=dcis,dc=uohyd,dc=ernet,dc=kvdi,dc=com
cn: sai
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
gidNumber:1001
givenName: Sai
userPassword: {SSHA}ywXiUa5tcTRUshEVfejflnr5V5FdOxDO
sn: V
homeDirectory: /opt/apache-tomcat-7.0.30/webapps/ROOT/IMPORT/10.5.0.96/home/sai/
loginShell: /bin/bash
uid: sai
uidNumber: 3002
```

Figure B.1: Example of ".ldif" file

- The syntax of *ldapadd* is :

  ldapadd -x -D cn=admin,dc=dcis,dc=uohyd,dc=ernet,dc=kvdi,dc=com -W -f user.ldif

- To generate the crypt password we use *slappasswd* command.

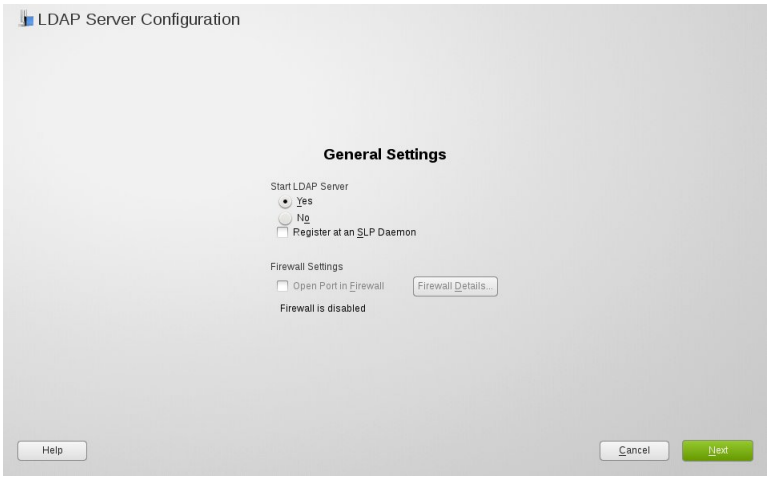

Figure B.2: (a)Ldap Server configuration file

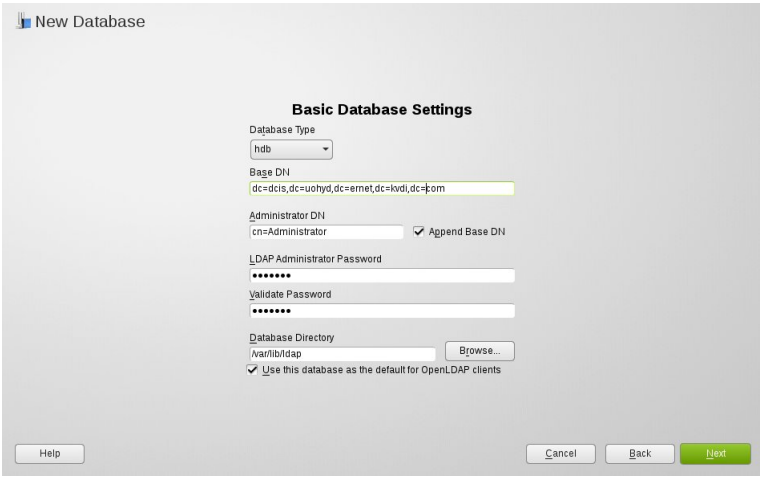

Figure B.3: (b)Ldap Server configuration file

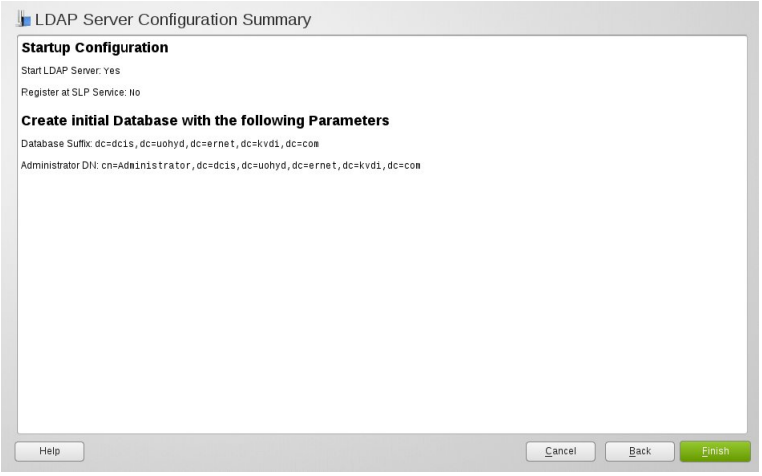

Figure B.4: (c)Ldap Server configuration file

B.3.3 Checking the LDAP Server

To check whether the LDAP server is installed and configured correctly, we need to search an entry (any user entry) in the LDAP server from any other system which is in the same network.

B.3.3.1 Searching Users in LDAP Server

- Use *ldapsearch* command to search any user in the LDAP server.
- The syntax of *ldapsearch* is :

  ldapsearch -x -h LDAP-Server-IP -p 389 -b cn=admin,dc=dcis,dc=uohyd,dc=ernet,dc=kvdi,dc=com "(uid=username)"

  e.g. ldapsearch -x -h 10.5.0.94 -p 389 -b cn=admin,dc=dcis,dc=uohyd,dc=ernet,dc=kvdi,dc=com "(uid=sai)"

- The sample output for a search is shown in below.

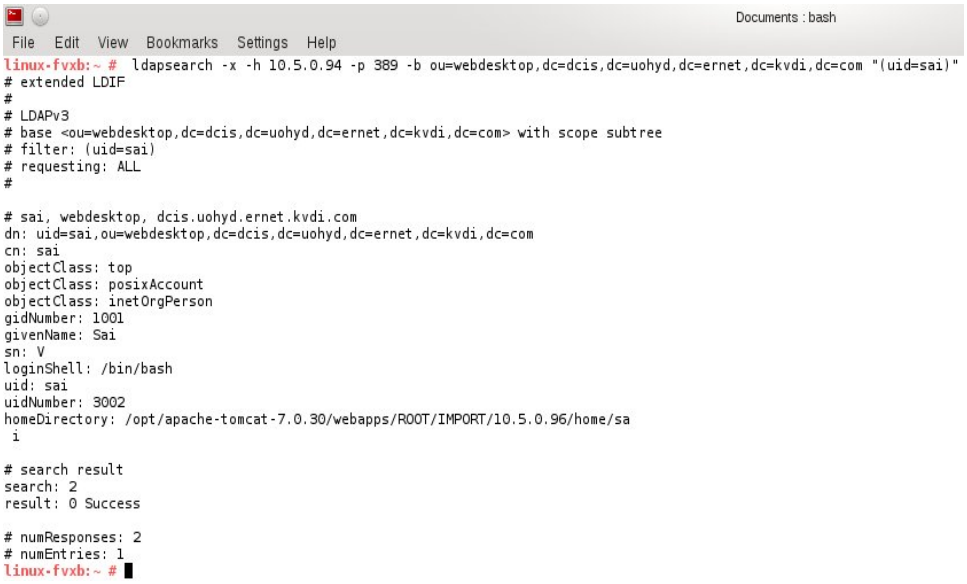

Figure B.5: Searching for a user in LDAP Server

B.4 NFS Server Installation and Configuration

NFS, or Network File System, is a client-server protocol for sharing files between computers on a common network. NFS enables you to mount a file system on a remote computer as if it were local to your own system. You can then directly access any of the files on that remote file system. The client system just needs to be running an NFS client compatible with the NFS server. This software needs to be installed, where the "user's home accounts".

In order to set up the NFS server you need to install the following packages:

- **nfs-kernel-server** - Linux NFS Server
- **nfs-common** - NFS Common programs
- **portmap** - The RPC portmapper

The installation will be done by GUI(software management in yast2). Once the installation is done, we need to export the required user's home directories which are going to be mounted in the web server. To do this, we need to configure the file */etc/exports*.

- Write the configuration line into */etc/exports* file by using following syntax

  Exported-directory-path ClientIP(options)

e.g. /export/home webserverIP(rw)

- "/export/home" is the exporting directory(user's home directory)
- rw:read and write permissions.
- We use web server IP as Client IP, so that web server is the only system which will access the user's home directories.

Then using the command $exportfs\ -a$ we can export the specified directories to the NFS client (web server).

In this system, the *ssh* daemon also should be enabled to access the applications. To enable the *ssh* daemon use the command: **service sshd start**. To know the status of *ssh* daemon use the command: **service sshd status**

Here, the web server itself is a nfs client. After nfs server is ready to export the user's home directories, the nfs client(web server) should mount those directories into it. For this, do the remaining configuration described in section B.2.2.1 .

B.5 Application Server Configuration

An application server should hosts all the required application. This not a necessary component in $SNAPWebD$ product. If a user does not have any application in his/her remote system, he/she should get the applications from application server. To get these applications, we need to install NFS Server in Application Server. This installation can be done as described in section B.2.2.2.
After the installation export the $/usr$ directory.

B.6 Client configuration

A client should accept the exported display from a remote server. For this check your client system's listen process with "$ps\ -ef \mid grep\ -i\ listen$" command. If the result have "$-no\ listen$" option, the system in not allowing the remote display. So to accept the display the client need to accept the xserver display. For this, we need to configure "/etc/sysconfig/displaymanager" file as.

- **DISPLAYMANAGER_XSERVER_TCP_PORT_6000_OPEN="no"**
  to
  DISPLAYMANAGER_XSERVER_TCP_PORT_6000_OPEN="yes"

By default the client system does not accept the remote server's display(xserver's display). Inorder to accept the display of remote server(NFS Server) run the command: $xhost +$.