

**Jukka Kommeri**

**System management in Server  
Based Computing with  
virtualization**

Helsinki University of Technology  
Department of Electrical and Communications Engineering  
Networking Laboratory

HELSINKI UNIVERSITY  
OF TECHNOLOGY

ABSTRACT OF THE  
MASTER'S THESIS

<b>Author:</b>	Jukka Kommeri
<b>Title:</b>	System management in Server Based Computing with virtualization
<b>Date:</b>	2nd October, 2008
	<b>Number of pages:</b> 51
<b>Department:</b>	Electrical and Communications engineering
<b>Professorship:</b>	S-38
<b>Supervisor:</b>	Professor Jörg Ott
<b>Instructor:</b>	Tapio Niemi, PhD, Helsinki Institute of Physics

The purpose of this thesis is to study the use of virtualization in server-based computing services. The results of this study were used to produce a prototype for the Finnish National Technology Agency (TEKES) funded NETGATE 2 research project. The prototype makes the installation of remote machines automatic and provides a way to distribute centrally managed services.

The installation and configuration of different services need a lot of work and knowledge. Many times this is repetitive as the services need to be installed in several locations and repeatedly. In a large organization this forms a significant expense. On the other hand small organizations cannot always afford an administrator and the administrating work falls to the person most capable. This results in insecure and unreliable installations. For these companies it would be beneficial to have the services ready to use without any expertise in the matter.

We solved the above problem by creating a system to distribute centrally managed services. This was done by enclosing generic services with their operating systems into transportable system images. These images can then be run on top of a virtualization platform as virtual machines. Images can be centrally created and maintained and then reused in various locations. Virtualization provides the images with a standard platform on which they can be placed.

The prototype uses the Xen virtualization and the Debian package management system to manage a collection of virtual machine images and distribute them according to the needs of the remote machines. The system gives the tools for automated deployment of the base operating system and virtualization tools over Internet. It also contains the tools for automated updates. In addition the system decreases the amount of administrative work and removes the need for expertise from the remote location.

Our prototype makes installation and management of large systems possible with less work than conventional systems. The system has been tested in several locations such the CERN library. The Debian package management system has proven to be an easy way to start up virtualized services and Xen has been good choice for the virtualization.

**Keywords:** SBC, virtualization, Xen, LTSP

<b>Tekijä:</b>	Jukka Kommeri
<b>Tyon nimi:</b>	Palvelun hallinta palvelin keskeisessä ympäristössä käyttäen virtualisointia
<b>Päivämäärä:</b>	2. lokakuuta 2008 <b>Sivuja:</b> 51
<b>Osasto:</b>	Sähkö ja tietoliikennetekniikan osasto
<b>Professori:</b>	S-38
<b>Työn valvoja:</b>	Professori Jörg Ott
<b>Työn ohjaaja:</b>	Tapio Niemi, FT, Helsinki Institute of Physics
<p>Tämän diplomityön tarkoitus on tutkia virtualisoinnin käyttöä palvelin-pohjaisen laskennan palveluissa. Tutkimuksen tuloksia hyödynnettiin luotaessa prototyyppiä TEKES-rahoitteiseen NETGATE-2 tutkimusprojektiin. Prototyyppi automatisoi etäkoneiden asentamista ja mahdollistaa keskitetysti ylläpidettyjen palvelujen jakelun.</p> <p>Erilaisten palveluiden asentaminen ja konfigurointi edellyttää paljon työtä ja tietämystä. Usein tämä työ on toisteista, kun samaa palvelua asennetaan useaan kohteeseen ja useaan otteeseen. Isossa organisaatiossa toisteisuudesta voi syntyä suurikin kustannuserä. Pienemmissä organisaatioissa eivät resurssit puolestaan aina riitä omaan ylläpitoon tai työtä tekemään valitulla ei välttämättä ole sopivaa koulutusta. Tästä on todennäköisesti seurauksena turvaton ja epäluotettava järjestelmä. Molemmissa tapauksissa olisi hyödyllistä voida käyttää keskitettyä palvelua joko omalta tai ulkopuoliselta ylläpidolta.</p> <p>Ratkaisimme ongelman kehittämällä järjestelmän, joka jakaa keskitetysti hallittuja palveluita. Tämä on toteutettu sulkemalla palvelut ja niiden käyttöjärjestelmät siirrettäviin järjestelmälevykuviin. Levykuvia voidaan ajaa virtualisointi-alustan päällä virtuaalikoneina. Keskitetysti luotuja ja ylläpidettyjä levykuvia voidaan uudelleen käyttää monessa kohteessa. Virtualisoinnilla tarjotaan levykuvissa oleville käyttöjärjestelmille vakio-alusta, jolla toimia ja näin taata toimivuus.</p> <p>Prototyyppi käyttää Xen-virtualisointia ja Debianin pakettienhallintajärjestelmää virtuaalikonekokoelman hallintaan ja jakamiseen etäkoneisiin. Järjestelmä mahdollistaa peruskäyttöjärjestelmän, virtualisointityökalujen ja automatisoidun päivitysjärjestelmän etäasennuksen.</p> <p>Prototyypimme tekee suurten järjestelmien asentamisen ja hallinnan kevyemmäksi kuin perinteiset menetelmät. Järjestelmää on testattu useassa kohteessa kuten CERN:in kirjastossa. Debianin paketinhallintajärjestelmä on osoittautunut helpoksi tavaksi käynnistää virtualisoituja palveluita ja Xen on puolestaan ollut tehokas virtualisointialusta.</p>	
<b>Avainsanat:</b>	SBC, virtualisointi, xen, LTSP

# Acknowledgements

This thesis was done for the Helsinki Institute of Physics Technology Program at Cern in Switzerland.

I would like to thank the Helsinki Institute of Physics and my boss MSc Miika Tuisku for making my stay in the scenic region of Switzerland possible. Also I would like to thank my instructor PhD Tapio Niemi, who kicked me forward and coloured many versions of my thesis. Thanks also to my Professor Jörg Ott for supervising my thesis and giving valuable comments.

Finally, I would like to thank my family for all encouragement and support given during my studies.

CERN Geneva, 2nd October, 2008

Jukka Kommeri

# Table of Contents

<b>Contents</b>	<b>vi</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>Abbreviations</b>	<b>ix</b>
<b>Abbreviations</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Problem Definition</b>	<b>4</b>
2.1 Netgate 2 . . . . .	4
2.2 Research Problem . . . . .	5
2.3 CERN Library Use-case . . . . .	6
<b>3 Server-Based Computing</b>	<b>7</b>
3.1 Thin Client Protocols . . . . .	8
3.1.1 ICA - Independent Computing Architecture . . . . .	9
3.1.2 RDP - Remote Desktop Protocol . . . . .	9
3.1.3 VNC - Virtual Network Computing . . . . .	10
3.1.4 X Window System . . . . .	10
3.1.5 NX - New X . . . . .	11
3.2 Linux Terminal Server Project - LTSP . . . . .	12
3.3 Summary . . . . .	13
<b>4 Virtualization and System Management</b>	<b>14</b>
4.1 Overview . . . . .	15
4.2 Benefits . . . . .	16
4.3 Methodology . . . . .	16
4.3.1 Paravirtualization . . . . .	17
4.3.2 Software Virtualization . . . . .	18

4.3.3	Hardware Support . . . . .	18
4.4	Xen . . . . .	19
4.5	VMware . . . . .	20
4.6	KVM - Kernel Virtual Machine . . . . .	20
4.7	Linux VServer . . . . .	22
4.8	System Management . . . . .	22
4.8.1	Planetlab . . . . .	22
4.8.2	Smart Domains . . . . .	24
4.9	Summary . . . . .	25
<b>5</b>	<b>Solution</b>	<b>26</b>
5.1	Debian Package Management System . . . . .	27
5.2	SBC Service . . . . .	29
5.3	System Components . . . . .	30
5.3.1	Client Installation Media . . . . .	30
5.3.2	Image Server . . . . .	31
5.3.3	Image . . . . .	31
5.3.4	Configuration Server . . . . .	32
5.3.5	Package Manager . . . . .	32
5.4	Operation of the System . . . . .	33
5.5	CERN Library Pilot . . . . .	34
<b>6</b>	<b>Evaluation</b>	<b>37</b>
6.1	Key Features of Prototype . . . . .	37
6.2	Usability . . . . .	38
6.3	Performance . . . . .	38
6.3.1	Distribution of Services . . . . .	38
6.3.2	Security and Reliability . . . . .	41
6.3.3	Virtualization . . . . .	42
6.4	Distributing Virtualized SBC . . . . .	43
6.5	CERN Library Use-case . . . . .	44
6.5.1	Performance . . . . .	44
6.5.2	Terminal Devices . . . . .	46
6.5.3	Open Source . . . . .	47
6.6	Discussion . . . . .	47
<b>7</b>	<b>Conclusions</b>	<b>49</b>
<b>A</b>	<b>Remote management system setup manual</b>	<b>52</b>
<b>B</b>	<b>CERN library thin client system</b>	<b>63</b>
	<b>Bibliography</b>	<b>67</b>

# List of Figures

1.1	Server-based computing . . . . .	1
3.1	Thin client and server roles . . . . .	8
3.2	Simple X window system architecture . . . . .	11
3.3	NX architecture (Courtesy of [36]) . . . . .	12
4.1	Traditional virtualization architecture . . . . .	15
4.2	x86 ring model (Courtesy of [2]) . . . . .	17
4.3	Xen hypervisor architecture (Courtesy of [28]) . . . . .	20
4.4	Kernel Virtual Machine architecture (Courtesy of [22]) . . . . .	21
4.5	Node management interface with the overall picture of node distribution . . . . .	23
5.1	Distribution system overview . . . . .	27
5.2	Virtualized SBC environment . . . . .	30
5.3	Workflow of the system . . . . .	34
5.4	CERN library pilot . . . . .	35
6.1	Network traffic between the thin client and the server in Bytes/second using X Window system . . . . .	42
6.2	Thin client used in CERN library (Courtesy of Gadget Computer)	46
7.1	Future vision of server-based computing . . . . .	50
B.1	The network of thin client system . . . . .	64

# List of Tables

6.1	SCP transfer speeds to VMs in MB/s . . . . .	39
6.2	Local transfer speeds in VMs in MB/s . . . . .	40
6.3	Workstation (WS) set up times . . . . .	45



# Abbreviations

ABI	Application Binary Interface
APT	Advanced Packaging Tool
CERN	l'Organisation Europ�enne pour la Recherche Nucl�aire
FS	File System
GDI	Graphics Device Interface
GUI	Graphical User Interface
HIP	Helsinki Institute of Physics
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
ICA	Independent Computing Architecture
KVM	Kernel Virtual Machine
LAN	Local Area Network
LTSP	Linux Terminal Server Project
MyPLC	My Planet Lab Central
Netgate	Network Identity, Grid Service Access and Telecom enabled provision
NX	New X
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
PXE	Preboot Execution Environment
RFB	Remote Frame Buffer

RDP	Remote Desktop Protocol
RMI	Remote Method Invocation
SBC	Server-Based Computing
SCP	Secure Copy
SSH	Secure Shell
TEKES	Teknologian kehittämiskeskus (Finnish Technology Agency)
TLS	Transport Layer Security
USB	Universal Serial Bus
VMM	Virtual Machine Monitor
VNC	Virtual Network Computing
WAN	Wide Area Network
WS	Workstation

# Chapter 1

## Introduction

Server-based computing (SBC) is an architecture where all the applications are installed, managed and executed on the server [5]. The computers of the users are considered as thin clients that exchange keyboard, mouse and screen information with the server. In pure SBC no actual programs are executed on the client machine. In the simplest case the client machine only runs the software required to transmit the user input and display the graphics.

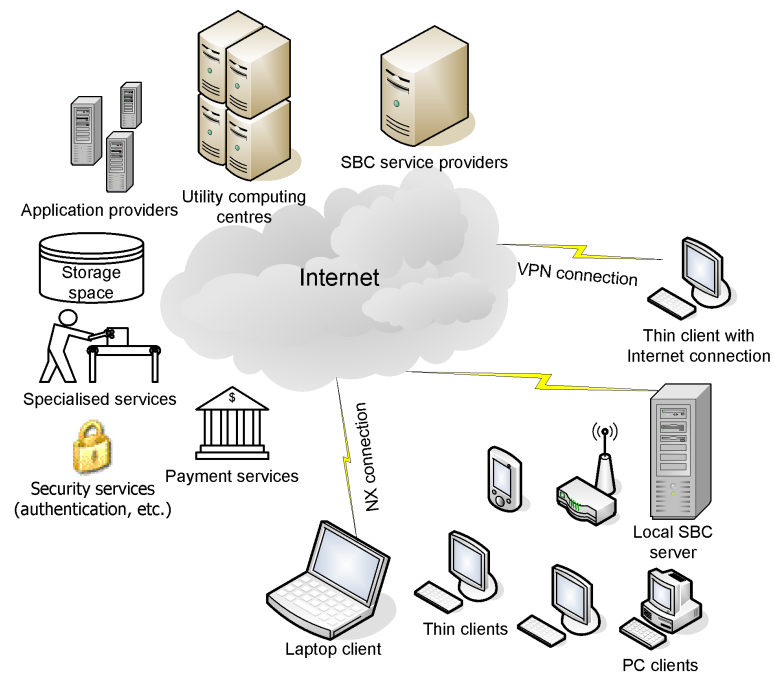


Figure 1.1: Server-based computing

SBC itself has been around for many decades [44]. It has been a good solution for many cases but not a complete solution due to its dependency on network speed and computing power on the server side. Today the computing power is quite cheap and the networks are getting faster. Now SBC can be considered as an actual alternative for the personal computer (PC).

SBC systems are easier to administer than PC-based systems because everything is on one machine, the server [33]. Adding new terminals requires no installation work as the client side devices are stateless. The hardware requirements for client side devices are minimal. Basically old PCs with network cards are enough. The centralization also increases the overall security since only administrators are allowed to configure the system and install new components.

There are several cases where big companies have adopted SBC technology. For example, in 2005 the Finnish Weather Forecasting Institute, Ilmatieteen laitos, announced to replace half of its desktops with thin clients. The University of Tampere has been using SBC for many years. There are several commercial SBC solutions such as the Sun Microsystems' Sun Ray [6], HP's server-based computing solution [20] and a newcomer, Panologic <sup>1</sup>, as well as non commercial ones such as LTSP <sup>2</sup>, K12LTSP <sup>3</sup>. According to Gartner's studies [25] the thin client market has increased 38% in the year 2006 which is according to them the highest since the year 2000.

One problem is that one SBC server can only serve a limited number of users and its setup requires a lot of knowledge. In a big organization more than one server is needed. The bigger the organization is, the more work is required to maintain the servers. To make it easier to handle multiple servers and reduce the end point administration, a central service distributor is needed: The server for servers that can be used to set up the SBC system in geographically dispersed sites and make the administration work more scalable.

This thesis studies the use of virtualization and remote installation systems to distribute and manage services like SBC on remote machines. Virtualization provides a standard abstraction on top of which different services can be installed. With virtualization it is possible to put the same service on top of different

---

<sup>1</sup><http://www.panologic.com>

<sup>2</sup><http://www.ltsp.org>

<sup>3</sup><http://www.K12LTSP.org>

physical hardware and replicate to various locations.

The thesis is structured as follows: In Chapter 2 we will look into different SBC technologies and define what SBC is and go through different implementations. After that, some virtualization and distributed system management technologies will be covered. Following these related work chapters, the actual solution and prototype will be explained in Chapter 5. Its properties will be further analyzed in the following evaluation chapter. Finally, we will summarize the thesis with the conclusion chapter.

# Chapter 2

## Problem Definition

In this chapter we take a look at the Netgate 2 project and the research problem of this thesis. The study was conducted at the Helsinki Institute of Physics (HIP) technology program premises at CERN. A working pilot of the Netgate 2 project is based on the solution of this thesis.

### 2.1 Netgate 2

Netgate 2 <sup>1</sup> is a research project funded by the Finnish National Technology Agency (TEKES) and Finnish industrial partners. The project is carried out and coordinated by Helsinki Institute of Physics in cooperation with Technology Business Research Center of Lappeenranta Technical University and the Department of Computer Sciences of the University of Tampere.

The purpose of the project is to study grid technologies and find ways for their adoption in Finnish industry. The project has three focus areas: security, server-based computing, and grid business research. This thesis is a part of NETGATE 2 and concentrates on the server-based computing.

The server-based computing part of the project studies how to simplify the installation and management of remote systems: Making it possible for a nontechnical person to set up complicated services in his premises and also relieve him from the administration. This decreases the overall administration costs and increases

---

<sup>1</sup><http://tek.hip.fi/opencms/opencms/projects/finnish/index.html#netgate-II>

the quality of administration for the end systems, i.e. ,the new installations are more robust and secure.

## 2.2 Research Problem

Installing servers and services assume a lot of understanding on the subject. In many cases the administrator is not fully qualified to do the installation but is selected for the task since he has the best knowledge about the subject [42]. This can lead to unstable and insecure systems. To be sure that the system works reliably it needs to be installed and administered by professionals. A problem is that the usage of professionals or having your own administration is expensive.

A out of the box solution is needed to make the services available in working environments that lack the required skills:

- A solution that makes the outsourcing of administration possible.
- A solution that makes possible the distribution of services such as SBC, firewalls, web servers, file servers, etc.
- These services would be configured and tested by professionals before introduction.

Tested services can be reused in several locations reducing the total amount of administration work. Clients can pick the set of services that they need for their environment and then just start their servers. The servers download all the required services from the central repository and launch them automatically.

The objective of this thesis is to study the SBC technologies and create a system that makes the distribution of SBC services less exhausting for the administration. We study the use of virtualization and automated installation for this problem and implement a prototype system that address the problem using only open source components. The suitability of this prototype will be tested at the CERN library.

## 2.3 CERN Library Use-case

CERN is the world's largest particle physics laboratory <sup>1</sup>. It is situated on the border of France and Switzerland just next to Geneva. It employs nearly 3000 employees and hosts some 6500 visiting scientists yearly. The mission of the CERN library is to acquire and manage information resources in all fields of relevance to the organization and make them easily accessible.

The CERN library provides its users with public terminals. These machines are mainly used to browse the Internet or to write documents. Currently desktop computers are used. They have Windows XP installed and connect to the CERN Windows domain. The installation of one of these machines takes about an hour and requires the involvement of the librarians. From the authors experience this procedure has to be repeated regularly, otherwise the machines become slower. Maintaining computers is not part of the core knowledge of the librarians and it interferes with their normal responsibilities.

The library is a good place to pilot the solution of this thesis. Replacing public terminals of the CERN library with a thin client system has many benefits. Thin clients are silent and easy to maintain. Thin clients do not need any installation as they work out of the box. They just need to be put into their place, connected to network and powered on. It takes about a minute for a thin client to boot and after that it is ready to be used. The installation of the server takes more time but with our system it can also be done automatically. All system components are open source, which gives us freedom to make more customized solutions that fit better to the actual need.

Thin clients and remotely managed server almost completely relieve the librarians from the administration of their public terminals. New thin clients can be added just by appending them to the same network as the others and by powering them up. Thin clients have a much longer life span than normal PCs, which in part reduces the administration work. The replacement of the desktop PCs with completely silent thin clients makes the library sound like a library.

---

<sup>1</sup><http://public.web.cern.ch/public>



# Chapter 3

## Server-Based Computing

Local area networks (LAN) and wide area networks (WAN) make it possible to centralize services. Instead of having all files and programs on local computers, as in the era of single user personal computers (PC), they can now be located on central servers and made reachable by others.

Centralization is not only limited to storage services but it can also be used to run programs. Due to the development of the networks and the exponential growth of processing power, the servers can now serve an increasing number of client applications. The benefits of this transition from local execution to centralized one is decreased administration work and increased system security. The same data can be accessed from several locations which gives more freedom for choosing the place to work at and does not make the loss of personal machine so critical. The server-based computing system resembles the old mainframe systems that were used in the early times of computers. The difference being that now the servers are a lot smaller and offer users more usable user interfaces [37]. The idea behind server-based computing is to move computation from PCs to centralized computing resources. The only things, the PC does, are:

- Maintain connection to the server.
- Display graphics.
- Transmit information from peripherals such as keyboard, mouse or USB device.

Thin client [5] is the client side device in the server-based computing. Almost

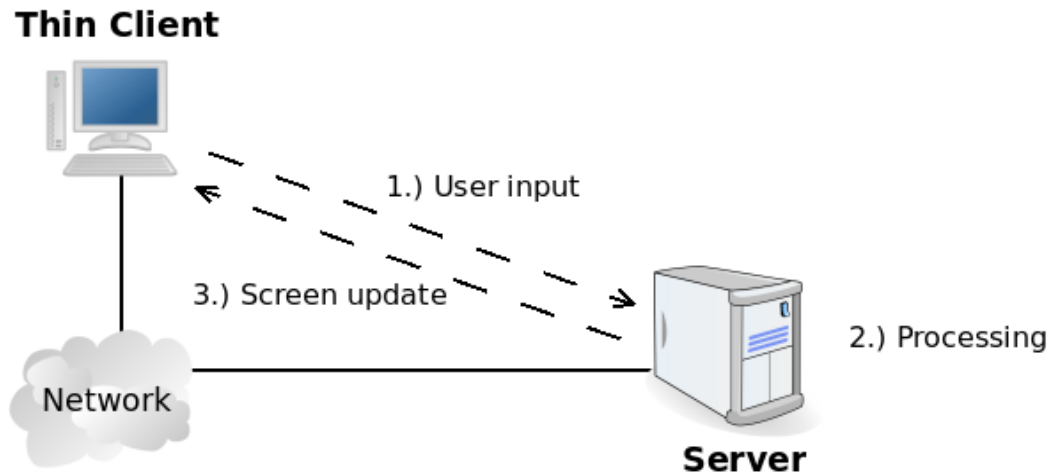


Figure 3.1: Thin client and server roles

any PC can be used as a thin client. The requirements for thin clients are low since most or all of the computation is done elsewhere. Thin clients do not need hard drives, powerful processors or powerful graphics cards. Having less powerful components means less energy consumption and less heat production. The ecological footprint of the thin client in the SBC system is about half compared to that of the PC[4]. The life cycle of the old and obsolete PC can be extended using it as the thin client. The requirements for a thin client vary on how much of the software is executed on remote servers [17]. In pure SBC all the programs are run on the remote server. It is also possible to enhance the user experience by running some of the programs on the client side.

### 3.1 Thin Client Protocols

There are several commercial products and open source projects that implement server-based computing. Most of them have their own protocols for transmitting information with optimized algorithms to meet the requirements of different networks and purposes.

Thin client protocols balance between the load on the network and the load on the end point machines: Usage of higher level graphics decreases the need to transmit data over the network but it requires more computation to form the image from the high level graphic primitive commands. The usage of raw pixel

encoding causes more network traffic but reduces the computation on the client side. The need to support the higher level graphics instead makes the client side more difficult to implement. Some protocols also adapt to the network traffic by sending updates less frequently, using caches, and compression [52, 26]. There has been a lot of development in the field of thin client protocols. Every new version brings performance improvements and features. This and the proprietary nature of some protocols make the comparison of the protocols difficult without empirical studies. In the following sub-sections we will introduce a few popular protocols that are used to provide server based computing.

### **3.1.1 ICA - Independent Computing Architecture**

The independent computing architecture (ICA) [23] is a proprietary protocol owned by Citrix. Citrix is one of SBC pioneers and offers multiple commercial solutions for both Windows and UNIX platforms. The SBC of Citrix is based on the ICA protocol though they also offer web-based solutions [39]. ICA is a lightweight protocol that can be used in low bandwidth networks such as modem connections. It can be used to share both Windows and Unix-based operating systems. A low level of network traffic is achieved by using higher level graphics primitives and compression of network traffic. ICA also allows the client to cache bitmaps which decreases the network traffic even more [52].

### **3.1.2 RDP - Remote Desktop Protocol**

The remote desktop protocol (RDP) [3] is a proprietary protocol owned by Microsoft. RDP is an extension to the ITU-T T.120 application sharing protocol family. It is used to connect to Windows Terminal Services running on Windows Servers. RDP sessions can be configured to meet various needs. It supports different levels of encryption. The traffic between the client and server can be secured with the transport layer security (TLS) [32] and compression is also supported. RDP uses higher level graphics. The server sends clients rendering data that the client uses to make API calls to the graphics device interface (GDI). RDP also supports roaming disconnect which means that the session stays alive though the connection dies [31].

RDP clients are made for Windows platforms. There are also open source RDP clients such as Rdesktop <sup>1</sup> and open source RDP servers such as Xrdp <sup>2</sup>.

### 3.1.3 VNC - Virtual Network Computing

VNC is an open source thin client protocol. It is based on the remote frame buffer (RFB). RFB is a thin client protocol that works on the frame buffer level making it applicable to all windowing systems and applications. VNC clients are stateless, which makes them tolerant to network disruptions. Both VNC and RFB have originally been developed at Olivetti & Oracle Research Laboratory [38]. VNC represents the 2D graphics and raw graphics end of the protocols. Clients retrieve pixel information from the server. This makes the clients easier to implement and they do not need a heavy system such as the X Window System to work. Therefore, VNC can be used in many light devices such as mobile phones and personal digital assistants [52]. The original VNC protocol did not support compression or encryption but they have been added to some derived products such as TightVNC <sup>3</sup>. Compression decreases the latencies of VNC in low bandwidth networks [24].

### 3.1.4 X Window System

The X Window System [34, 43] is the base for graphical user interfaces (GUI) in many Unix compatible operating systems. It implements the client server model. Applications are clients that connect to the server that handles input from keyboard and mouse and output to monitor. Clients can be on the same computer as the server or on a remote one. Both client and server use X protocol for their communication.

Figure 3.2 illustrates how the X server, that runs on a workstation, is connected to two X clients, that are on two separate remote machines, and also to two local X clients.

The X window system is designed for local area networks. Though it uses high

---

<sup>1</sup><http://www.rdesktop.org>

<sup>2</sup><http://xrdp.sourceforge.net>

<sup>3</sup><http://www.tightvnc.com>

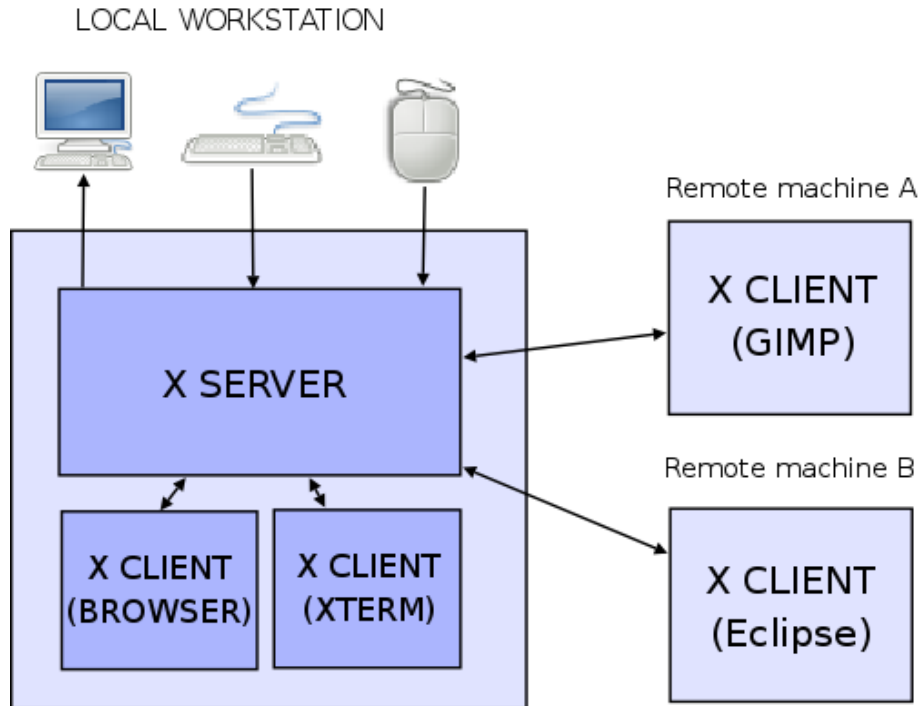


Figure 3.2: Simple X window system architecture

level graphics, it sends updates more frequently than others and does not use any compression, which makes it one of the heaviest protocol on the network. These properties also make it the best quality protocol [52].

### 3.1.5 NX - New X

New X (NX) [36] is a protocol that enhances the basic X Window System making it more usable in wide area networks. It acts as a buffer between the client and the server of the X Window System. It can store the session information on the remote machine making the X Window System more resilient to network errors. NX provides an encrypted and compressed link between the client and the server machines. The protocol can also be adjusted to meet different network connections by suppressing the updates of the X Window System. Figure 3.3 illustrates the architecture of the NX protocol.

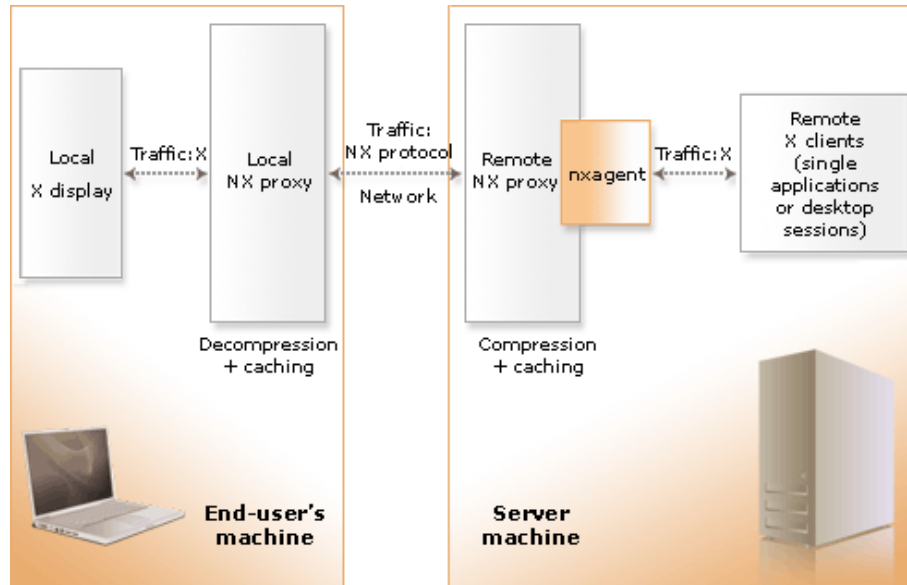


Figure 3.3: NX architecture (Courtesy of [36])

## 3.2 Linux Terminal Server Project - LTSP

The Linux Terminal Server Project was founded in 1999 and has since then been included in various Linux distributions. It has also been the base for the foundation of a new Linux distributions such as K12LTSP <sup>4</sup>. At the time of writing of this thesis LTSP is in version 5.0 with the codename MueKow <sup>5</sup>. LTSP is an open source project that combines many other open source projects into a comprehensive SBC solution. It offers various ways to connect thin and fat clients to the Linux server from different networks. The main usage of LTSP is to connect thin clients in the local area network to the central server using the X Window system.

In LTSP, one does not need to install any operating systems to the workstations. They are booted with small Linux that is loaded either by PXE boot or from local media such as floppy, CD-Rom or USB stick. No hard disks are used at the client side, which makes it possible to boot a desktop machine with an operating system as a thin client without any harm to the existing system. All tools required for managing the thin client and its graphical sessions are loaded from a network drive using the network file system (NFS) or the network block device (NBD).

<sup>4</sup>[www.k12ltsp.org](http://www.k12ltsp.org)

<sup>5</sup>[wiki.ltsp.org/twiki/bin/view/Ltsp/MueKow](http://wiki.ltsp.org/twiki/bin/view/Ltsp/MueKow)

The thin client only runs X server and connects to a session manager on the server in the LAN [30].

### 3.3 Summary

Server-based computing is an architecture in which most of the programs and personal files are executed and stored centrally. Centralization makes the resources accessible from various locations and with different devices providing users with flexibility.

The server and client communicate using a middleware. The middleware passes the input of the user to the server and the graphical output of the server back to the client. Most of today's thin client middleware offer the same functionality. The choice of a protocol depends much on the use case: Is it going to be used in the local area network or over the Internet and is the end device a computer or a mobile phone. The X Window system offers good quality for free but exhausts the network. ICA and RDP are proprietary protocols that are made to work well on low bandwidth networks. VNC is the easiest one to implement and it can be made to work on a very simple devices such as the mobile phone but it does not offer very good quality.

# Chapter 4

## Virtualization and System Management

In this chapter we take a look at virtualization: What it is and why its usage would be beneficial. After the overview, we introduce a few remote management systems that employ virtualization.

Virtualization is a widely used term. Basically, it refers to the abstraction of physical and other resources. Virtualization offers standard interfaces for applications and operating systems and removes their dependency on the underlying hardware or software layer. Hardware resources can be multiplexed between several operating systems and made to look like something else using virtualization. It is possible to virtualize complete machines or just parts of the machine.

A good example of virtualization is the Java programming language. Java code is compiled against the Java environment and runs on top of the Java virtual machine. The Java virtual machine always looks the same for the code independent of the hardware. This makes it possible to execute the same code on top of many different hardware architectures.

In this chapter we concentrate on the virtualization techniques that are common with the x86 architecture. To be more precise we concentrate on the platform virtualization: Techniques that make it possible to run several operating systems on top of one set of hardware.



## 4.1 Overview

Virtualization techniques have been around for many decades. They were introduced in the era of mainframes. Then, the virtualization was used to multiplex the scarce resources among multiple applications. Nowadays one use of virtualization is to decrease the proliferation of server machines and to improve their cost efficiency. Not only does it reduce the requirement for hardware but it also reduces running costs such as electricity and space [41].

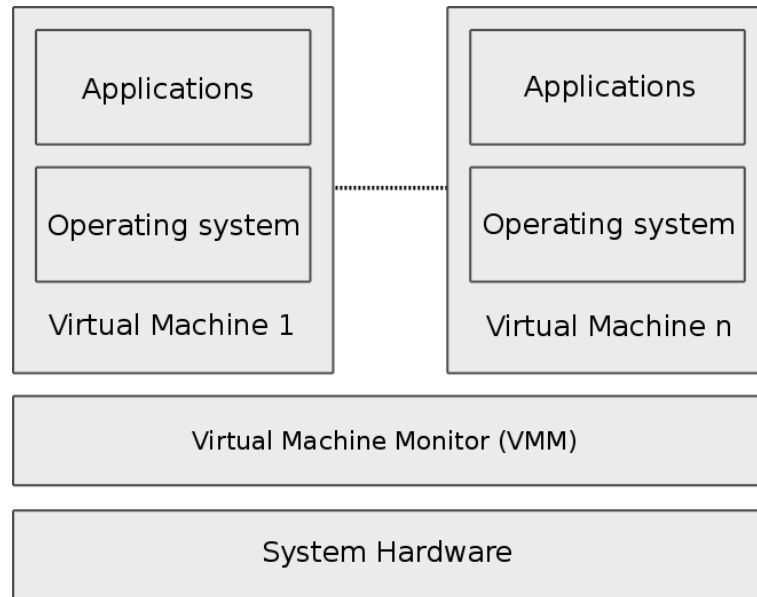


Figure 4.1: Traditional virtualization architecture

With virtualization, it is possible to divide the physical hardware among several virtual machines (VM). These VMs are controlled by a virtual machine monitor (VMM). VMM is the abstraction layer that hides the hardware below and provides a generic interface for the virtual machines. Figure 4.1 illustrates the virtualization architecture. Applications and operating systems run as they were run on the physical hardware. VMs are isolated from each other by VMM so that they cannot affect each other [40].

VMM can reside either on top of an running operating system or directly on top of hardware. Running VMM on top of an operating system introduces a lot of overhead but also makes setting up of a testing environment easier.

## 4.2 Benefits

Virtualization allows us to enclose applications to their own environments. Having applications on separate machines increases the overall security of the system. The more applications or services there are on one machine, the more insecure the machine becomes. If one service is compromised then the others are as well. For example an intruder may use a security hole in one service to gain administrative privileges on that machine. Placing applications into their own virtual machines creates a protective barrier between them [51].

To have better fault tolerance, the applications should be distributed among several either physical or virtual machines. Having multiple applications on the same machine increases the risk of them affecting each other. One crashing or updating may interfere the others. These interferences can lead to unnecessary service down times.

The need for fault tolerance and inexpensive hardware has led to the proliferation of hardware. Most of them running idle 90% of the time. Virtualization provides a more cost efficient solution. Instead of having several idle physical machines one can have a few better utilized machines populated with virtual machines [41].

The encapsulation of operating systems and applications into virtual machines makes them also more movable. It eases the set up and migrations of services into new locations. One only needs to install the virtual machine monitor on the new machines and then copy existing virtual machines on top of it. This cuts down the installation times and provides a way to run several versions of the same software in parallel. With virtualization it is possible to set up testing, development and production versions of the same system on one machine.

## 4.3 Methodology

The x86 architecture is not designed to support virtualization. This is due to the privilege levels of the instructions of the processor. Figure 4.3 illustrates the ring structure of the x86 architecture. The level 0 also known as kernel mode is able to execute all instructions and level 3, guest mode, a subset of this. The right to execute certain instructions depends on which level the instruction was executed

[13].

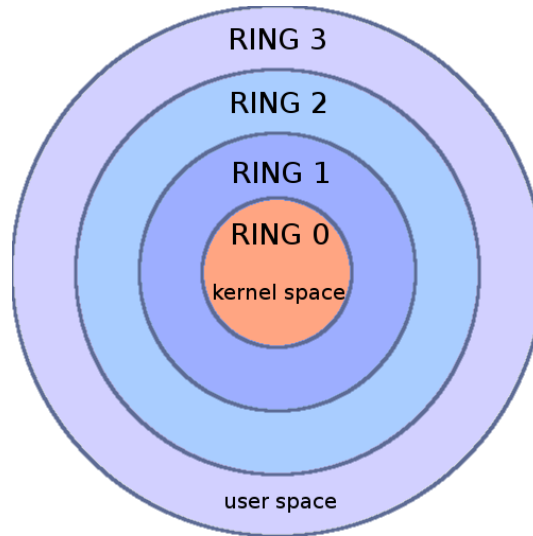


Figure 4.2: x86 ring model (Courtesy of [2])

Normally operating systems operate at the level 0 and applications at the level 3. In virtualized environment the virtual machine monitor works at the level 0 and guest operating systems at higher level. Without any virtualization support the virtual machine running on the higher level fails to execute any privileged instruction [47].

There are a few solutions to the deficiencies of the x86 architecture. Here we introduce three main categories which are used in most of the current virtualization solutions: 1) Paravirtualization, 2) Software virtualization, and 3) Hardware virtualization.

### 4.3.1 Paravirtualization

The architecture used with paravirtualization was originally developed by IBM and utilized in the VM operating system[14] <sup>1</sup>. The term paravirtualization was introduced at 2001 [50] by the Denali group <sup>2</sup>. Since then paravirtualization has gained much popularity and is now used by leading virtualization developers such as XenSource and VMWare.

---

<sup>1</sup><http://www.vm.ibm.com/>

<sup>2</sup><http://denali.cs.washington.edu/>

There are many different implementations of the paravirtualization technique but the main idea remains the same. Paravirtualization provides the guest operating system a hardware abstraction that is similar but not an exact copy of the underlying hardware. The use of paravirtualization requires modifications to the guest operating systems. These modifications reduce the complexity of the virtual machine monitor and enhance its performance. The original machine instructions are either modified or excluded. Privileged commands are made to communicate with virtual machine monitor [51].

A problem with paravirtualization is that the owners of proprietary operating systems might not be willing to modify their OSs. Some OS instructions need to be modified for the virtual machine to able to operate on the lower privilege level [19]. The patching of any existing operating system requires a lot of knowledge and work. This complicates the selection of the OS flavor and makes the other virtualization methods more viable [49].

### 4.3.2 Software Virtualization

Software virtualization contains a set of virtualization techniques that are used to provide full virtualization [40]. Full virtualization means that there is no need to change the operating system above. Names of the techniques vary depending on the level they are used at.

Binary translation is one form of software virtualization and it is used to emulate machine instructions. Instructions that are executed by the operating system of the virtual machine are translated from the source instruction set to the target instruction set [46]. This form of virtualization does not require any changes to the virtual operating system. It is a form of virtualization where the VMM reforms VM's commands for the underlying hardware. The translation of the commands naturally introduces some overhead but can it also optimize and gain performance boost with some instructions [8].

### 4.3.3 Hardware Support

Due to the rise of interest in the virtualization, processor manufacturers have included virtualization support in their products. Intel VT Vanderpool/Virtual

Technology [47] has separate support for 32-bit and 64-bit architectures. VT-x for the 32-bit IA-32 architecture and VT-i for the 64-bit Itanium architecture. AMD's Pacifica chip [53] includes AMD-V virtualization support .

Both Intel and AMD versions of hardware support try to solve the problems of the x86 architecture, described above, by adding a separate mode for the guest OS. In this mode the guest OS is able to run on the ring level 0 and execute privileged commands in a normal way [47, 53].

## 4.4 Xen

Xen started as an open source project at the University of Cambridge. Later it was productized by XenSource<sup>3</sup>. Now there are both commercial and open source versions<sup>4</sup> of the project. Xen has been included into various Linux distributions such as Ubuntu, Suse, and Red Hat. Xen VMM has also been included into the Linux kernel. Xen comes with a vast set of tools from the creation of the virtual machines to the live remote migration.

Xen uses paravirtualization as its virtualization technique. Its VMM is called the hypervisor. The architectural structure of the Xen hypervisor is illustrated in Figure 4.3. The idea behind the Xen hypervisor has been to keep it as small as possible. Much of the management and control features have been moved to the privileged guest domain called domain0 [29].

Domain0 is brought up at boot time and it is able see all the hardware. It contains hardware device drivers and has tools to manage guest operating systems, so called virtual machines. Guest operating systems are given an abstraction of the device drivers, the front end pieces. Front-end pieces communicate with their corresponding back-end drivers that are int the domain0. Moving the device drivers from the VMM to the dom0 lightens the hypervisor and also gives protection against faulty drivers [9, 29].

Though Xen requires modifications to the guest operating system, no changes are needed to the application binary interface (ABI). This makes it possible to run the guest applications unmodified[9].

---

<sup>3</sup><http://www.xensource.com>

<sup>4</sup><http://www.xen.org>

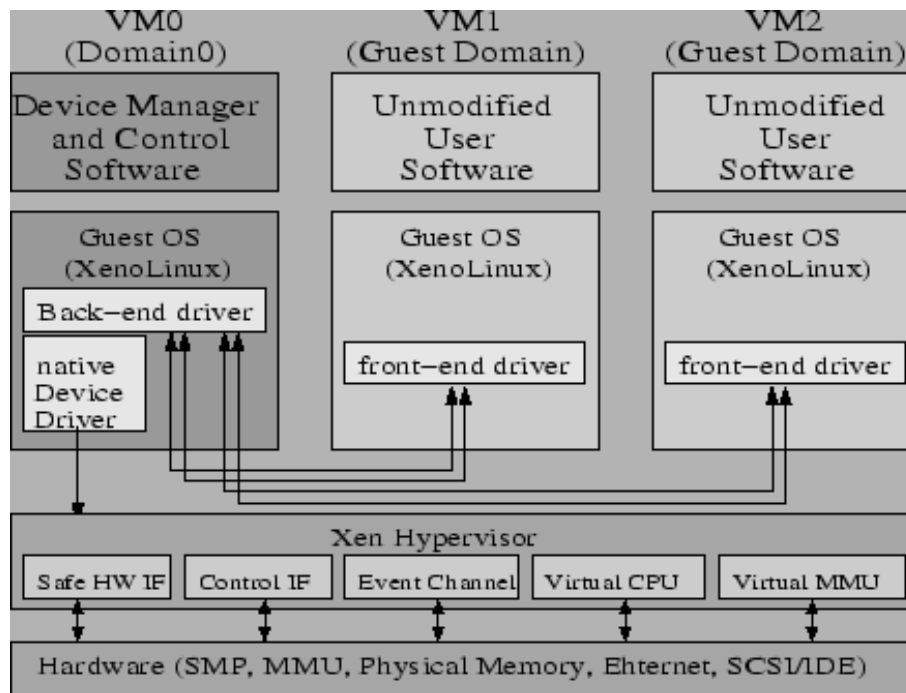


Figure 4.3: Xen hypervisor architecture (Courtesy of [28])

## 4.5 VMware

VMware is one of the biggest virtualization solution providers. It offers a wide range of virtualization products from desktops to servers. Desktop virtualization products such as Workstation are used on top of an existing operating system as a normal application [45]. Server virtualization, such as VMware ESX, works on top of the bare hardware <sup>5</sup>.

Most of the products of VMware are proprietary but it also offers some open source products. VMware uses mainly binary translation in its products but also supports paravirtualization [7].

## 4.6 KVM - Kernel Virtual Machine

Kernel Virtual Machine (KVM) is a newcomer in the virtualization domain [35]. It exploits the recent hardware virtualization enhancements in processors, which

<sup>5</sup><http://www.vmware.com/products/esxi/>

were described in Subsection 4.3.3. It has been included in the Linux kernel <sup>6</sup> since the version 2.6.20.

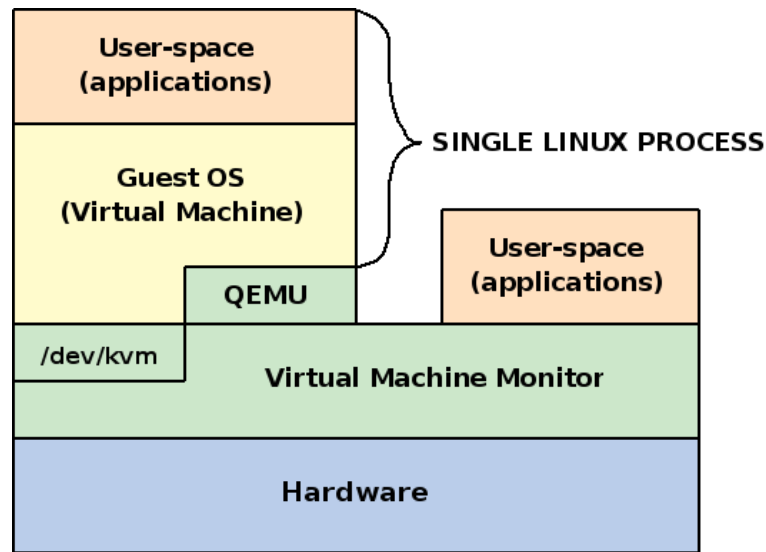


Figure 4.4: Kernel Virtual Machine architecture (Courtesy of [22])

Figure 4.4 illustrates the architecture of KVM. The Linux kernel is turned into a virtual machine monitor by adding a KVM module into it. Virtual machines are processes on top of the host operating system. This means that KVM uses the scheduling and the memory management properties of the vastly developed Linux kernel.

The KVM module enables near machine speed virtualization of the processor and the virtualization of memory inside kernel. I/O devices of the guest operating system are handled with a user space emulation tool QEMU <sup>7</sup>. QEMU emulates the I/O for guest operating systems and provides virtualization of I/O devices.

Two main disadvantages of this otherwise simple and powerful virtualization technique, are the emulation of I/O devices in user space and the need for special hardware. Especially the emulation of I/O slows down otherwise fast hardware virtualization [22].

<sup>6</sup><http://www.kernel.org>

<sup>7</sup><http://bellard.org/qemu/qemu-doc.html>

## 4.7 Linux VServer

Linux VServer <sup>8</sup> is another virtualization technique that takes advantage of properties built into the Linux kernel. Virtual machines are run in the user space as normal programs and they share the operating Linux kernel with the underlying operating system and other virtual servers. All processes of virtual machines are controlled by the same kernel.

Vserver virtual machines contain their own root and security context. This means that the virtual machines are unaware of other virtual servers or the underlying operating system. Though the file system of the VM is a part of the file system of the underlying operating system, the VM has no way of accessing or seeing anything beyond its file system. Separate security context means that the virtual machines only see their own processes [15].

## 4.8 System Management

Virtualization is used by remote installation and management systems to provide tools for resource abstraction and multiplexing. The two following subsections show how the virtualization is used for the management of remote resources and how it provides a standard interface for services.

### 4.8.1 Planetlab

Planetlab is an overlay network of computers. In the early 2008, the Planetlab network consisted of more than 800 computing nodes in over 400 geographically separated sites <sup>9</sup> in every continent. This range of distribution provides an excellent testbed for the network or distributed computing research.

To become a member of the Planetlab network one must donate computing nodes to the network. Figure 4.5 illustrates how the Planetlab network scatters around the world to cover all the continents. Dots on the map represent computing nodes of the members of the network.

---

<sup>8</sup><http://linux-vserver.org/>

<sup>9</sup>[www.planet-lab.org](http://www.planet-lab.org)



Home

**Slice hip\_gb - Nodes**

Select a site to add nodes from.

CESNET – Czech Education and Research Network

planetlab1.cesnet.cz

planetlab2.cesnet.cz

planetlab1.fit.vutbr.cz

Add Nodes

---

**Nodes currently associated with slice**

Check boxes of nodes to remove:

planetlab1.diku.dk

planetlab2.diku.dk

planetlab1.hiit.fi

planetlab2.hiit.fi

host2.planetlab.informatik.tu-darmstadt.de

host3.planetlab.informatik.tu-darmstadt.de

planetlab-1.man.poznan.pl

planetlab-2.man.poznan.pl

planetlab1lannion.elibel.tm.fr

planetlab2lannion.elibel.tm.fr

planetlab1.cslab.ece.ntua.gr

planetlab2.cslab.ece.ntua.gr

Remove Nodes

[Back to Slice](#)

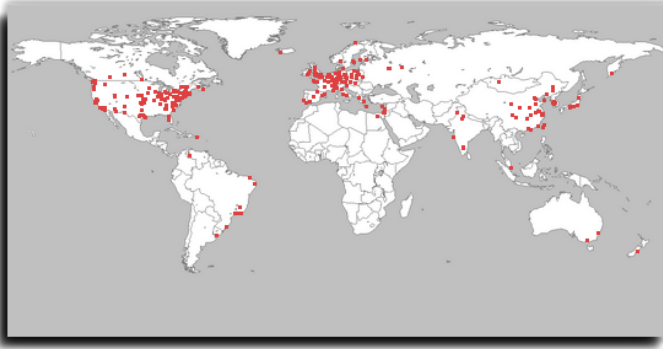


Figure 4.5: Node management interface with the overall picture of node distribution

Planetlab has several projects that are called slices. In practice slices are sets of virtual resources. One node can contain several slices. A slice on a node means that the owner of the slice has an active virtual machine running on that node. A slice can have an arbitrary amount of computing nodes from anywhere in the world. The virtualization on the node machines is implemented with the Vserver virtualization technique.

The management of slices and nodes is centralized. They are created via the Planetlab central server putting the final control in the hands of Planetlab administration [12]. The centralized control also means that the management actions are initiated by the central server.

The Planetlab provides users with processing power from machines that are geographically dispersed. These virtual machines contain basic operating system tools. Additional software has to be installed by the user. The Planetlab com-

munity provides some tools for the management of software on the slices <sup>10</sup>.

### **My Planetlab Central - MyPLC**

My Planetlab Central (MyPLC) is the software used in Planetlab. With MyPLC is possible to set up an own overlay network. The code of the MyPLC code is freely usable, allowing people to make their own versions of the Planetlab software [1].

### **4.8.2 Smart Domains**

Smart Domains is a virtual machine resource management infrastructure. It is designed to set up distributed environments for running batch GRID jobs and conducting system tests. Smart Domains builds on the Smartfrog framework and Xen virtualization. Smartfrog provides a way to configure, deploy, monitor, and manage large distributed environments [18].

Smartfrog is a Java-based software made by HP labs <sup>11</sup>. Smartfrog provides:

- A rich description language for describing resources and their interoperability.
- A deployment engine for delivering configurations and software to the peer to peer network of Smartfrog nodes.

Smartfrog uses Java remote machines invocation (RMI) for communication between peer to peer nodes. RMI makes Smartfrog less applicable in large scale networks as firewalls usually block RMI traffic [16].

In Smart Domains, the management of virtual resources is done by submitting resource descriptions to the network. All work is automated from the creation to the destruction of resources by the framework.

---

<sup>10</sup><https://www.planet-lab.org/tools>

<sup>11</sup><http://www.hpl.hp.com/>

## 4.9 Summary

There are many virtualization solutions for platform virtualization. They all have their applications and their suitability depends much on the requirements. The requirements can vary from the setting up testing environments on a PC to the consolidation of a room full of servers.

Picking a suitable virtualization solution depends on many things such as cost, easiness of adaptation, efficiency, technical limitations, and security. Proprietary virtualization solutions are made easy to use and they offer good support. Open source products require more knowledge to set up but they are free and customizable. User space virtualization solutions such as KVM and VServer offer powerful virtualization solutions that are easy set up but depend on the functioning of the shared kernel. Paravirtualization and binary translation are both used to provide a foundation for virtual machines with complete operating systems. These VMs have normal access to I/O devices with unnoticeable overhead but setting them up is more difficult and every virtual machine introduces overhead in the use of memory.

# Chapter 5

## Solution

The goal of the thesis is to make a system to set up and replicate SBC services at geographically distributed locations. This can be achieved by using virtualization. Virtualization offers several benefits. For example virtualized services can be easily moved to different locations and started on top of the abstraction provided by virtualization.

We have developed a system that distributes services inside virtual machines. All the installations using our system have the same base operating system and virtual machine monitor. These machines are then complemented with virtual machines that contain the actual services such as web-server, print server etc. This is visualized in Figure 5.1.

Virtualization offers a way to enclose services into their own specific and optimized environments. Having services in separate virtual machines makes the system structure more modular and manageable. One server can be used to host several virtual machines. Enclosed and separated the services do not disturb each other and updating one does not cause the others to suffer any disturbances. Separation also adds a layer of security between different services, which means that compromising one service leaves others intact.

We use Ubuntu <sup>1</sup> as the operating system. Both virtual machines and the virtualization platform use Ubuntu. All software can be found in Ubuntu Edgy and subsequent distributions, which means that all used components are open source.

---

<sup>1</sup>[www.ubuntu.com](http://www.ubuntu.com)

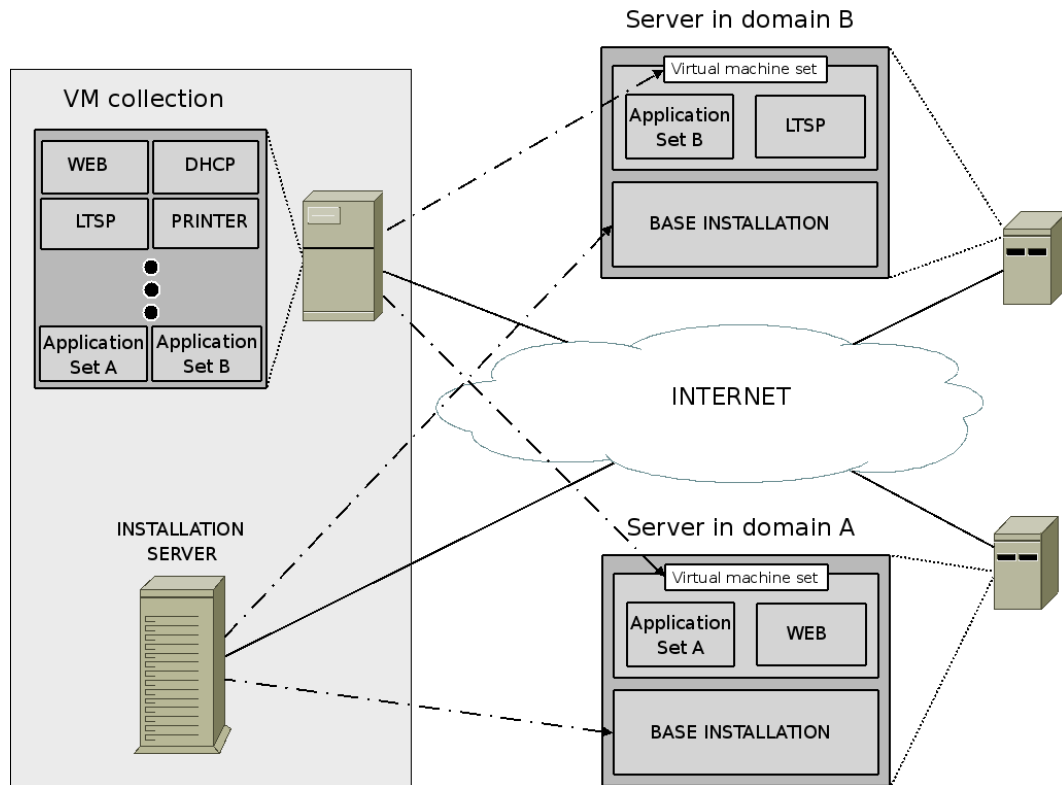


Figure 5.1: Distribution system overview

Since we have mostly used existing software, there is little code to upkeep.

As our virtualization platform we have chosen Xen and paravirtualization. Xen is open source and it has improved quite a lot in the past few years. The paravirtualization of Xen offers good performance with little overhead. Paravirtualization does not use any emulation so its performance does not decrease in I/O intensive use. Xen also supports hardware virtualization but does not require it, which makes it suitable for heterogeneous set of machines.

## 5.1 Debian Package Management System

The Debian package manager is a powerful software management system and the foundation of many Debian-<sup>2</sup>based operating systems [10]. It can be used to install, remove, and upgrade software. Managed software, either in binary

<sup>2</sup>[www.debian.org](http://www.debian.org)

or source form, is stored and distributed as compressed packages. The package manager can use both external or local package sources.

The Debian package management system can be divided into three main building blocks. The first one is a package library also called a repository, which contains the software packages with different versions of them. The second component is the package in which the software is enclosed. The third part contains the client side package management tools. They are used to install, remove and update software on the client.

The repository is actually just a collection of files in a predefined directory structure. Normally, the structure is a tree where the distributions are at root level and categories such as main, non-free, and contrib as their subdirectories and below them separate folders for different architectures. In every leaf directory there are special files such as Packages.gz and Sources.gz depending on the contents of the directory. Sources.gz is used with source packages and Packages.gz with binary packages. These files contain the metadata of packages.

The repository can reside on a local media such CD-ROM and DVD or on a separate file server, which can be accessed with well known protocols such as FTP and HTTP. Distributions such as Debian or Ubuntu have repositories, which contain thousands of packages. Packages are categorized into different distribution releases by their readiness and compatibility. Distributions evolve and go through three phases of development from unstable to testing and from testing to stable. To protect users, repositories and their packages can be signed with a private key. These signed repositories and packages can then be verified with the public key by the user.

The package is the container for software. Besides storing the actual files for the software it also includes metadata and possibly installation and removal scripts. Packages can contain either pre-compiled binaries or source code. Metadata consists of information such as a version number, package name, author name, packages dependencies, etc. Package dependencies determine what other packages are required for the software in the package to work properly.

The Debian package management system has many tools for software installation, removal and upgrading. All the tools are build on top of the dpkg<sup>3</sup> (Debian

---

<sup>3</sup><http://en.wikipedia.org/wiki/Dpkg>

GNU/Linux Package Manager) tool. Dpkg is a powerful package management tool but it lacks the ability to handle dependencies. APT (advanced package tool) or its successor, the Aptitude<sup>4</sup>, can handle these better. APT and Aptitude can be configured to use multiple repositories with the `source.list` configuration file.

Package management operations are client-initiated. First, the client fetches all `Packages.gz` files from the repositories. Then, the metadata from these files are used to build package database on the client. When installing a new package, this database is searched for information on the installation candidate. If the package exists in the database, the actual package is fetched from the repository and installed.

## 5.2 SBC Service

In our system SBC services are installed into virtual machines. In this SBC system we have separated access and application server to their own virtual machines. The access server is responsible for setting up thin clients. It contains the LTSP environment described in Section 3.2. The application server has all applications and the desktop environment to which the thin clients connect to when they are set up. It is installed with the Gnome<sup>5</sup> desktop environment and some common office tools.

Figure 5.2 illustrates our virtualized SBC system. We have one physical machine that runs several virtual machines. There can be several versions of the same service running in parallel, which makes version updates as convenient as possible. In the figure the physical machine also works as a firewall. Virtual machines are constrained to their own LAN and they have the physical machine as the gateway. Depending on the hardware of the physical machine, the virtual LAN may also be connected to another physical LAN via other network card, making it possible to have the thin clients in their own network.

---

<sup>4</sup><http://wiki.debian.org/Aptitude>

<sup>5</sup>[www.gnome.org](http://www.gnome.org)

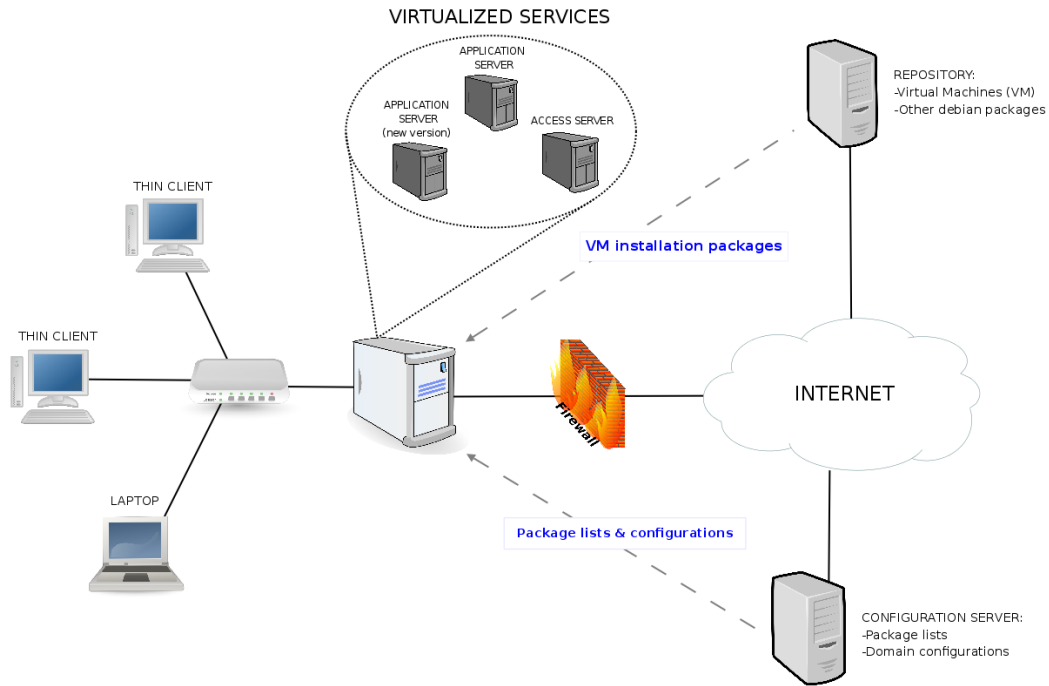


Figure 5.2: Virtualized SBC environment

## 5.3 System Components

In this section, we explain the components that comprise the automated installation and management system. The complete installation documentation is included in the appendix A.

### 5.3.1 Client Installation Media

The client installation medium is used to install remote machines with a base installation. The base installation contains a basic operating system, virtualization tools, and remote management tools. Booting a machine with the installation medium loads the Linux operating system that handles the installation of the client server. The installation will set up Ubuntu Feisty OS with the Xen virtual machine monitor. The installation is automated using the Debian Installer preseed files.

The preseed file contains the answers to the questions that Debian Installer normally asks on console. All configuration information on installing the machine



and getting installation packages etc. are included in the preseed file. A version of the preseed file, which is used in the NETGATE-2 project, can be found in the appendix A. Preseed files are stored and managed on the central server. Centrally located preseed files are easy to update and make the actual installation media more generic.

### 5.3.2 Image Server

The image server is basically a Debian/Ubuntu package repository, i.e., a file server that can be accessed with HTTP. It is used to store Debian packages. The repository contains both the actual packages and the metadata of the packages in a predefined tree directory structure. The metadata is used by Debian and Ubuntu installation tools, the apt-tools, to install and manage packages and their dependencies [21].

The repository itself is created and managed with an open source tool called Reprepro <sup>1</sup>. Reprepro manages the versioning of packages and it also signs every package when it is added to the repository. The public key of the repository, that is used to verify the packages, is retrieved during the base installation due the configurations in the preseed file.

### 5.3.3 Image

Virtual machines are distributed by using image files. One image contains a file system with a complete operating system root directory. Images are compressed into Debian packages, which significantly reduces the size of the image. A 4GB virtual machine image can compress to 270MB since the empty space in the image is completely compressed. The effectiveness of the compression makes the VM images more movable in the network.

Every VM has its own Debian package. Usage of Debian packages makes it easy to manage the files, that are required by the VM, and easy to script functionality for the installation process. Installation using Debian packages automatically sets up the required files to the file system of the client machine. Shell scripts are used to prepare the environment for VM and eventually to start the VM.

---

<sup>1</sup><http://mirrorer.alioth.debian.org/>

Separate scripts are made for stopping the VM and removing all the additions made by the installation scripts.

The tool for creating the installation packages is `dpkg`. `Dpkg` is given as parameter a directory that contains all files needed by the virtual machine in a predefined tree structure. When installing the image, all the files in it are placed on the root of the target machine according to the predefined tree structure. One exception is a special control folder which is in the root of the tree. This folder contains all the scripts and metadata of the image and is only used by the installation and removal tools such as `apt` and `aptitude`.

### 5.3.4 Configuration Server

The configuration server stores the installation package lists of the clients. Lists are distributed in pull fashion using the `Rsync` program <sup>6</sup>. `Rsync` is a remote transfer program that keeps track of changes in files and eliminates all unnecessary transfers. The clients periodically synchronize their lists with the configuration server.

Usage of pull strategy was chosen to enhance the overall security of the system. The more incoming ports are open the more insecure the system becomes [48]. Now the traffic is generated from inside and no extra holes are made to the firewall for the incoming traffic.

### 5.3.5 Package Manager

Package manager is an installation package management tool. It is installed as part of the base installation. The package manager retrieves package lists from the configuration server and updates the software of the machine accordingly. It uses `Rsync` and `SSH` to communicate with the server. Lists are retrieved on a daily basis and also at boot time.

After retrieving the lists the package manager calls `pkgsync` <sup>7</sup>, which is a package management tool of the operating system. `Pkgsync` synchronizes packages

---

<sup>6</sup><http://samba.anu.edu.au/rsync/>

<sup>7</sup><http://manpages.ubuntu.com/manpages/feisty/en/man8/pkgsync.html>

according to the list. Every package mentioned in the list is installed and missing packages are removed. For example, if a software package that was on the previous list is not on the current list, it is removed from the machine. Also if there are updates to the packages currently installed, they are applied.

## 5.4 Operation of the System

The work flow of the system in short is as illustrated in Figure 5.3:

1. The client machine installation is started using a generic installation medium.
2. Installation parameters are retrieved from a configuration server.
3. Installation packages are downloaded from a repository and installed.
4. The Package manager is started and a package list is retrieved from the configuration server.
5. Virtual machine installation packages are retrieved from the repository and installed according to the list.
6. Go back to step 4.

The client machines are installed using a remote installation service. The user is given a USB-stick or CD-ROM or any other medium, that can be used to boot computers from. This medium is used for the installation of the client server. All the user has to do is inserting the media into his computer and power up the computer. It will then start the installation, which runs automatically and will install the client machine with a base operating system and virtualization environment. In the current prototype, the system still asks a few questions but the goal is to make it completely automatic.

After the installation, the client computer retrieves a list of installation packages from the configuration server. The retrieval of the list is done periodically and this is the way the software contents are managed on the client server.

The list of installation packages is used to update the package collection of the client. If the list introduces new packages to the client package collection they will be downloaded from the repository. All packages that were installed on the client and are not on the new list will be removed. The list can contain packages from

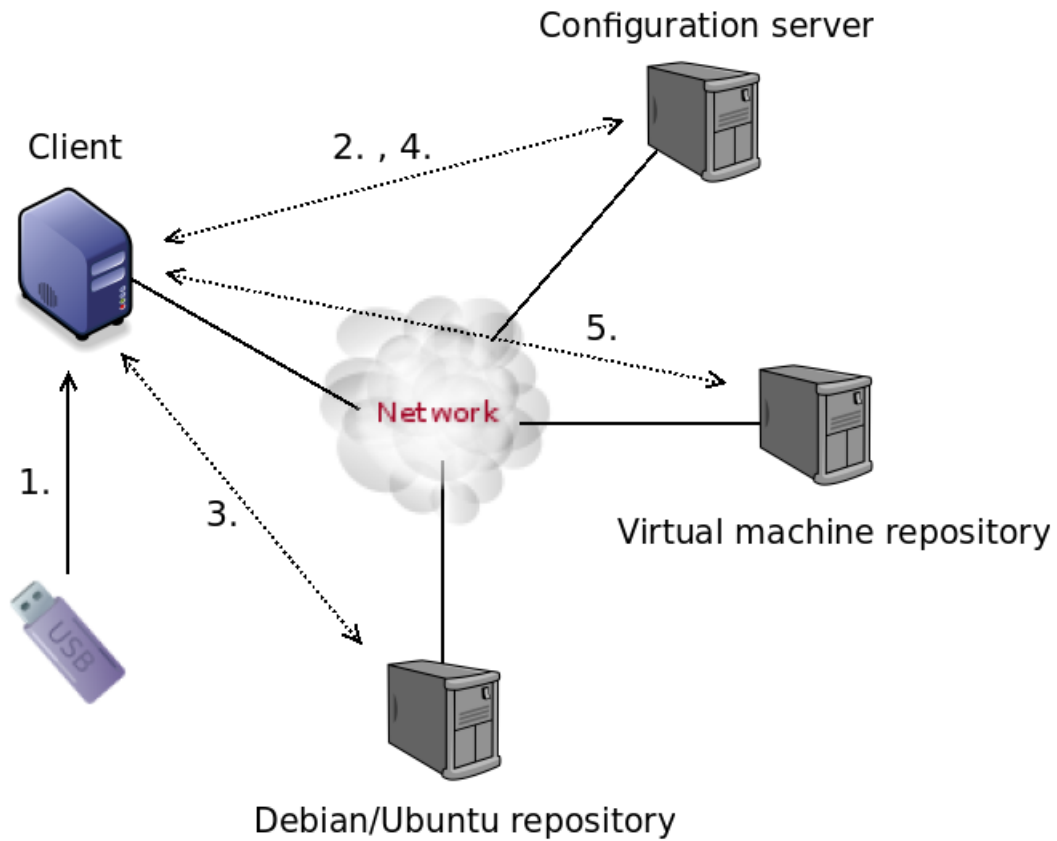


Figure 5.3: Workflow of the system

the repository of the base operating system and from our own virtual machine repository.

Package lists contain all software packages needed by the base installation. The lists also includes some additional packages that contain virtual machines destined for the machine. Installation of these packages causes the client server to launch new virtual machines and the removal of these packages causes the virtual machines to shutdown and remove all related files.

## 5.5 CERN Library Pilot

The CERN library needed an easily maintainable and silent system that fits their needs. The system described in this chapter is used to deliver a custom-built server-based computing system to the library. Updates to the systems are

automatic and the remotely managed. Figure 5.4 illustrates the architecture of the CERN library pilot.

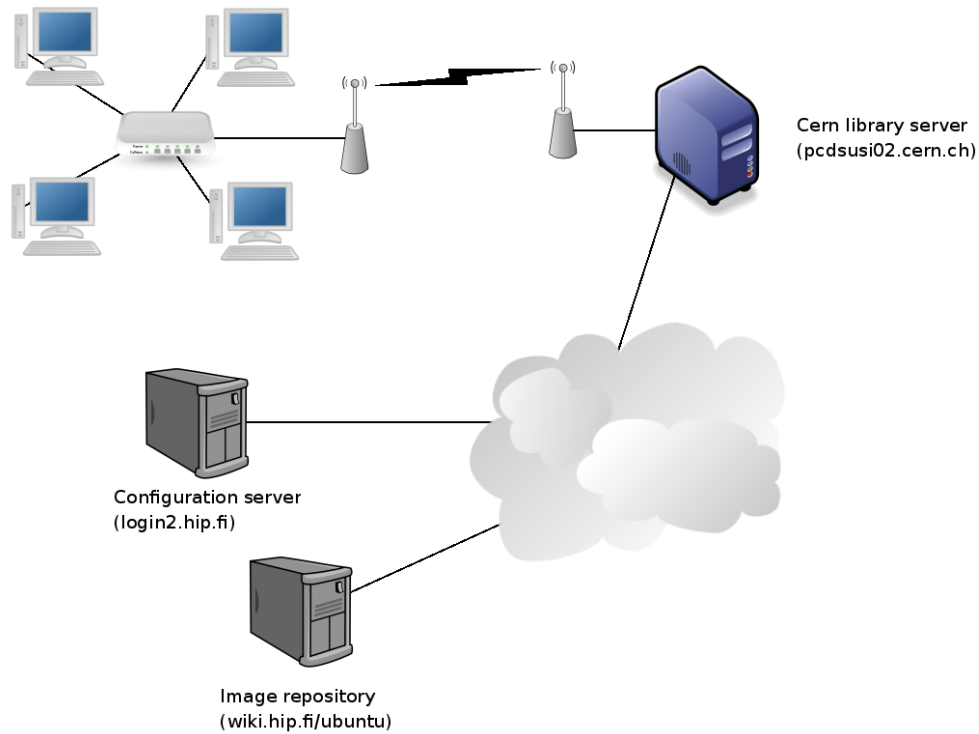


Figure 5.4: CERN library pilot

One of the old desktop PCs was chosen to be the server for the sbc environment of the library. The server was installed using our systems. The library also obtained some economical, diskless and power wise limited computers <sup>8</sup>, which would nowadays be considered useless for anything else. But as thin clients they work well enough.

The server hosts two virtual machines. One handling thin clients and one to provide desktops and applications for the thin clients. The VM that handles the thin client has the Ubuntu Feisty operating system and the LTSP environment is called access server. The other server, called application server, is also installed with Ubuntu Feisty on top of which we have installed the Xfce4 <sup>9</sup> desktop environment with all the required library software. For both virtual machines the applications were chosen carefully to minimize total system load. Communication between thin clients and the application server is done using the X window

<sup>8</sup>[http://www.icoptech.com/products\\_detail.asp?ProductID=271](http://www.icoptech.com/products_detail.asp?ProductID=271)

<sup>9</sup><http://www.xfce.org/>

system, which is by default very insecure. To provide users with a trustworthy environment the communication was encrypted using secure shell (SSH) tunnels.

To be able to place the thin clients more freely, the clients were connected to the server using wireless bridges. Wireless bridges makes the installation of cables simpler as now there is now need for an cable network between the server and clients. Clients can be connected to the server wirelessly. Figure 5.4 illustrates how we have connected a small Ethernet network of thin clients to a server using wireless connection. The thin clients and the server are in the same local area network as though they were connected with a wire. Having wireless connections between the server and client also introduces security problems which are now dealt with SSH encryption.

# Chapter 6

## Evaluation

In this chapter, we evaluate the solution described in the previous chapter. The technologies chosen for the solution are compared to other corresponding technologies. We will also analyze how well the solution implements the requirements set in the problem definition. Finally we will have a look at the benefits and challenges of our pilot project in the CERN library.

### 6.1 Key Features of Prototype

The solution introduced in this thesis solves the problem of distributing server-based computing services to geographically dispersed locations. Organizations with no specialized administration was considered as the main target group.

The problem was tackled by completely automating the system installation and management process. This was planned to achieve with a network installation system and with the virtualization of services.

In our solution the services are tested and packaged centrally. In this way, we are able to make the installations more reliable for the end user. This makes the end product more robust and secure as it would be if made by a person that started from scratch. Centralization also adds scalability as the same services can be used many times. The virtualization layer is used to add security and to make the setting up and removing of services more reliable. Virtualization provides services a standard foundation to be placed on.

## 6.2 Usability

All administrative tasks, such as the creation of virtual machines and configuration of client package collections, are handled centrally. The end user has almost no responsibilities and relies on the central administration. This arrangement fits well for the category of people to whom the system was designed for, though the setting does not restrict more experienced users to do their own modifications.

Clients have rights to modify their system but the changes are not applied to the installation images on the central server, which makes the changes temporary. This can be seen as a good feature as all possible damages that the client does can be unmade and the central administration has the final word.

The installation of the end system is automated using the old and matured Debian Installer. Answers to the questions the installer normally would ask are pre-answered by a preseed configuration file. This removes the need for any involvement in the installation process from the end user. After the user has put the installation medium into the machine and switched on the machine, the installation system should be able to do the rest.

## 6.3 Performance

The system relies heavily on high speed network connections, since the installation media and configurations files are located on remote servers. The client installation medium introduced in Section 5.3.1 only launches an installer that retrieves the installation media from remote repositories. It is possible to have a local repositories for images but their management would be more complex. Having a centralized server makes the management more scalable because the same configurations and images can be reused in multiple locations.

### 6.3.1 Distribution of Services

The virtual machines are distributed using image files. These files contain complete Linux root file systems (FS). The actual size of the image of one FS does not matter so much as the empty space compresses more than thousand-fold.



This makes the FS size indifferent and the transfer of the virtual machines with large file systems feasible.

There are several ways to provide the virtual machine FS [11]. One can use physical partitions, logical volume manager volumes (LVM), network file systems, or file backed disks such as loopback devices. In our solution we have chosen to use loopback devices as they can be prepared completely at the server side and easily set up at the client side.

The loopback device introduces some overhead because it is a file system on top of another file system. The relevance of this overhead is not significant and loopback devices perform relatively well. In our tests we noticed that loopback disk images perform at the application level as well as, or even better, than the actual physical disks.

Table 6.1: SCP transfer speeds to VMs in MB/s

SCP to asynchronous disks				
	Physical disk partition		Image	
	Big file write	Small files write	Big file write	Small files write
No load	5.91	0.21	11.91	0.21
Load	0.63	0.09	7.30	0.17
	Big file read	Small files read	Big file read	Small files read
	No load	28.09	0.22	27.47
Load	4.63	0.10	4.45	0.08
SCP to synchronous disks				
	Physical disk partition		Image	
	Big file write	Small files write	Big file write	Small files write
No load	5.64	0.20	13.06	0.20
Load	1.14	0.17	8.67	0.19
	Big file read	Small files read	Big file read	Small files read
	No load	23.51	0.22	26.27
Load	6.86	0.15	5.49	0.13

Tables 6.1 and 6.2 illustrate the transfer speeds gathered from our test environment. Environment had two almost identical virtual machine except that one had its FS on physical partition and the other on a loopback device image.

Table 6.2: Local transfer speeds in VMs in MB/s

Local transfers with asynchronous disks				
	Physical disk partition		Image	
	CP (MB/s)	DD (MB/s)	CP (MB/s)	DD (MB/s)
No Load:				
Big file	23.73	23.33	17.56	16.60
Small files	0.88	1.48	0.89	1.44
Load:				
Big file	5.11	6.09	5.02	4.59
Small files	0.18	1.30	0.17	1.22
Local transfers with synchronous disks				
	Physical disk partition		Image	
	CP (MB/s)	DD (MB/s)	CP (MB/s)	DD (MB/s)
No Load:				
Big file	2.39	0.43	6.76	1.93
Small files	0.39	1.50	0.43	1.44
Load:				
Big file	0.55	0.09	2.16	0.86
Small files	0.13	1.40	0.15	1.27

Table 6.2 illustrates the transfer speeds of cp and dd commands inside the virtual machines and inside one FS. Table 6.1 illustrates the read and write transfer speeds between a virtual machine and another server in the same local area network. The small files set were the same in all tests and consisted of 3890 files with average size of 7.9kB. Big files were different in the two cases, 1GB file was used in the scp tests and 500MB in the local tests. The tests were performed with default settings and no block size optimization was used with dd.

Both transfer speed tests were performed both with and without load in the domain0. The load was generated by running in parallel simple programs that stress the processor and also by having several parallel data transfers locally and over network. Also, the significance of the caching of FS was tested having separate tests for synchronous and asynchronous modes.

Our testing showed that loopback devices perform relatively well. Loop back device FS images do not depend on any other software as the LVM and the NFS do. Setting up and removing the image on the client side is quite simple. These features make image files suitable for the remote installation system.

### 6.3.2 Security and Reliability

The most important advantage of image files is their transportability, which makes the preparation and delivery of services more robust. The images can be prepared and tested thoroughly by professionals and then moved to remote locations for usage. Virtualization provides an interface for these prepared images and removes the need to do any installation or configuration operations at the remote site.

Centralized management also makes the services more secure as the work will be done by professionals with accumulated experience on the subject. The network of clients gather data that the centralized administration can then use to enhance services. This is a security issue as it is important that the programs are well configured and up to date.

Virtual machines are distributed using repositories. It is possible to eavesdrop the traffic between the client and server and taint the contents. This is why the packets in the repository are signed with the private key of the repository making it possible for the client to check the integrity of received packages.

The public key of the repository is fetched by the automated installation via HTTP making it insecure. It would be possible to use HTTPS for this but it would require the use of certificates to make the server trustworthy for the client. Having certificates in the installation media makes it less generic as the certificates need to be replaced regularly and also it is not, to our knowledge, supported by the Debian package management system.

### 6.3.3 Virtualization

Our system uses virtualization to host virtual machines. With virtualization it is possible to multiplex the resources of one machine among several virtual machines. This makes it possible to set up several services in one machine without affecting the others. Having services in their own machines creates a new layer of security as there is no actual link between machines except probably a network connection. Virtualization provides this security and reduces the need to purchase several physical machines and at the same time makes the use of existing resources more efficient.

Transforming several physical machines into virtualized ones and running them in one physical machine makes them more manageable. All virtualized machines can be turned on and off from one place. If virtualized machine seizes up it can be rebooted using the `dom0`. Virtualization and the consolidations make the physical host machine the single point of failure. This should be taken into consideration with better hardware or with the duplication of the hardware. Though this is the case with critical services even without virtualization.

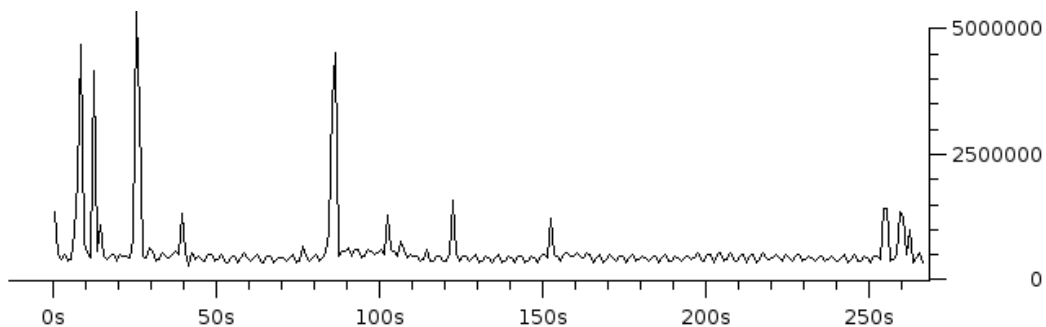


Figure 6.1: Network traffic between the thin client and the server in Bytes/second using X Window system

Among several different virtualization solutions, we chose to use the Xen project. Xen uses paravirtualization that offers us high performance in an open source package. With paravirtualization we get fast I/O performance which is not reduced by the emulation like most other virtualization solutions. Virtual machines do not run in the kernel space so the virtualization platform is quite well protected. These features make Xen a good virtualization product for our use as the SBC environment with the thin client communication produce a lot of I/O load. Figure 6.1 illustrates the traffic of one thin client in 100 MB/s local area

network. During the traffic capture time the thin client was used to browse the Internet and write a small document. As it can be seen, the amount of traffic one thin client produces is significant even when being idle.

The Xen project offers a multitude of tools for the management of virtual machines and pools of servers. VMWare has corresponding virtualization products but its licensing and proprietary nature makes it less appealing for our use as the plan is to use open source components. The tools provided by the Xen project can be better reused in the future development and the recent activity in the project promise more enhancements in the future.

Xen has also a support for hardware virtualization and thereby it can run non Linux operating systems such as Windows. Though, then it has to rely on emulation and suffers performance losses. As all the services needed in this project can be found in Linux distributions we have no need for hardware virtualization support. However it is good to have this possibility open for future development.

## 6.4 Distributing Virtualized SBC

So far the discussion in this chapter has been mostly about generic services. Now we will take a look at how the system serves server-based computing. SBC, as described in Chapter 3, is a system where all the applications, processing and storage space are centralized. It can be used to offer applications, operating systems, and access to devices all of which can be accessed via thin client protocols.

In our prototype the access service and the application service are separated into their own virtual machines. These virtual machines can easily be deployed anywhere with the virtualization environment.

In our basic set up we only have two virtual machines. One offers a booting media for thin clients in the LAN and the other provides a desktop environment with applications. Once these systems have been configured and tested in our environment they can be reused in other sites as well. The virtual machine repository and Debian packages have proven to be a very reliable way to automate the installation of virtual machines.

SBC comprises of different server oriented services. There are services such as

printing server, file server, proxy server, firewall etc. All these can be made into a virtual machine and served as module to the clients. Not all SBC services are generic and require a lot of configurations on site. Still these can be prepared well before hand so that the actual installation and configuration effort on site would stay minimal.

## 6.5 CERN Library Use-case

The CERN library is used as a test environment for our solution. The idea is that the system relieves the librarians from their IT administrative tasks and provides a usable and secure system for the visitors of the library. In this section our solution is compared against the existing alternative, the MS Windows workstations.

The CERN library is an easy case as it has no need to integrate existing services with ours. It only needs a terminal service for their customers. This system should provide the users with basic office tools and access to the Internet. The network connection of the CERN library is ideal: It has very little restrictions and offers 100MB/s bandwidth. This makes the installation and updates fast and also makes the server more usable as a gateway for thin clients.

### 6.5.1 Performance

We tested our implementation by comparing it with the CERN Windows installation system. While the installation of one workstation from the CERN Windows domain takes about 2.5 hours, the installation of one Xen server with virtual machines required by the thin client system takes about 50 minutes. The set up time of one thin client is about 5 minutes, which is the time it takes to set up the hardware. Table 6.3 contains some values gathered from our tests performed in our environment. The values shown are not absolute values but suggestive results of our simple testing from which one can see the time savings achieved.

Table 6.3: Workstation (WS) set up times

	SBC	Windows
Time components:		
HW installation / PC	5 min	5 min
Base system	10 min	45 min
Virtual machine	15 min	-
Office capability	35 min	1 h 35 min
Total time:		
Installing 1st WS	50 min	2 h 25 min
Setting up a new WS	5 min	2 h 25 min
Reinstalling services	15-40 min	-

Table 6.3 illustrates some key installation times. First there is the time component division. The total times consists of these components. In both SBC and Window workstation case the actual hardware installation time is considered the same as in both cases the same components such as wiring needs to be put into place. The Windows side has two time components: the first one is the minimum time it is needed to be able to use Windows. The office capability portion is the time that it takes to set up the additional software needed to perform basic office tasks. The latter part is done in the background so that the workstation can be used, but the background installation takes most of the capacity. In the SBC case the time needed to install the first terminal comes from the installation of the hardware of both the server and the thin client and the installation of the server virtualization platform with two virtual machines. The thin client itself does not require any additional installation than the placement of the hardware.

The difference in the initial set up times of these two systems is not that crucial as several Windows terminals can be installed in parallel. The time saving in the SBC system comes from the upgrades and re-installations. Both systems require updates from time to time. Some installations like security patches can be automated but sometimes there might be need to make complete re-installations. With thin client system only the server needs updating as with Windows every machine needs to be reinstalled. With our system it is enough to just update

the virtual machine serving the thin clients. This can be done remotely so that almost no downtime occurs.

### 6.5.2 Terminal Devices

Windows requires much more powerful terminals i.e. workstations than the SBC. Usually they also have a lot of wearing parts such as hard drives, fans etc. These parts have limited life span and need replacing in a few years. At some point these parts become unavailable, which forces the renewal of the whole workstation. Thin client can be just about any machine with a supported architecture. The machines used as thin clients have not been powerfull enough to work as a windows workstations for many years. These machines are made, with todays technology, very small and without any wearable parts.



Figure 6.2: Thin client used in CERN library (Courtesy of Gadget Computer)

Machines without any wearing parts have longer life span. The need to replace old terminals with newer ones comes seldom. Having no fans or disks also means that there are no noise producing parts in the machine. Windows with all necessary office applications needs a top of the line machine even to work. These machines produce a lot of heat, which means they need cooling. If the number of terminals or workstations were bigger then also the energy consumption would be an issue.



The thin clients consume one tenth of that of the normal workstation [27] making the SBC .

### 6.5.3 Open Source

Windows workstations offer little or no chances to modify the system to better suit the needs. With open source we have combined a system of only the required components and modified them to work in this special case and with the low end hardware. This way the system will not become bloated and too heavy to use. Also, it is easier to make the modifications that are requested. For example now the librarians have the possibility to modify the desktop appearance, of all terminals in a simple manner.

## 6.6 Discussion

The system is easy to build since it uses common tools from the Linux distributions. These tools have been around for many years and have had the time to mature. Though the installation media and software from the Ubuntu distributions tend to work quite reliably, it is not 100% reliable. There is always room for a human error when the security updates are released. Currently our system relies on the functioning of the distributions repositories but can be made the way that we have control over distribution updates. Having this kind of service of course adds more work and complexity to the management.

The same installation media can be used to set up a cluster of servers. These servers can contain auxiliary services for load balancing. With the virtualization one can have all kinds of different services on one physical machine. Clusters could be built the way that they can balance load of the services between servers. Xen offers tools for live migration of virtual machines. One could even retrieve the virtual machine by setting up a repository service. This would make the launch of a new service in the fast LAN even faster.

The images provided by our system can be customized for special needs but that makes them less generic and suitable for others needs. Images that are more generic can be applied easily to several locations. For some services this is

suitable. Our library thin client system for once. There they have no need for authentication or dedicated user accounts.

To get most out of the network installation and centrally prepared services, the services should be less user specific. The more customization there is in one service the less applicable it is in other places. Though, the installation system can be used to lighten the installation of services to some extent leaving the final customization for the local administration.

With images it is also quite easy to provide the client with encrypted file systems. This would restrict the local administrators of dom0 from accessing the image and its contents. Encryption could also be used to add security to the transfer of images from the server to the client as it makes the packages useless for intercepting parties. Though the traffic can also be secured using TLS, it makes the installation process more complicated.

Usage of centralized management of the images can be reasoned in many ways. One is that in a centralized system the know-how will grow in one place and mistakes wont be repeated in other places. This way the mistakes done in the early stages of the service lifetime can most likely be avoided.

Although it is quite handy to have the installation and management service centralized there is always the question of who will actually perform it and what is the price. As that subject is out of the scope of thesis it will not get any more attention here.

# Chapter 7

## Conclusions

In this thesis the server-based computing, virtualization, and techniques of service distribution were studied. The research was done as part of the Netgate 2 research project. As a result a system that automates the installation and maintenance of remote servers with virtualized services was produced. The system has been tested in several locations such as the CERN library.

The prototype uses virtualization to create small modular service packages that can be distributed among geographically dispersed locations. The system makes the setup and replication of distributed and complex services possible. The installation of a new remote machine has been made as easy as possible.

With virtualization one can package services into virtual machines and combine these into modular and secure systems. Most of the applications used in the prototype are in several Linux distributions. This includes also the Xen paravirtualization, which makes it a good solution as the virtualization platform. Paravirtualization induces less overhead than other virtualization solutions since it does not emulate or translate machine instructions. The Debian package management system was chosen for the distribution of virtualized services because it is a mature system and has proven to be functional. It offers all the necessary tools to distribute and manage virtual machines.

While writing this thesis, many possible applications for server-based computing have arisen. There are applications like language labs, office environments, tele working etc. One thing in common to all of them is centralization of resources and simplification of end devices. Figure 7.1 illustrates a visionary arrangement

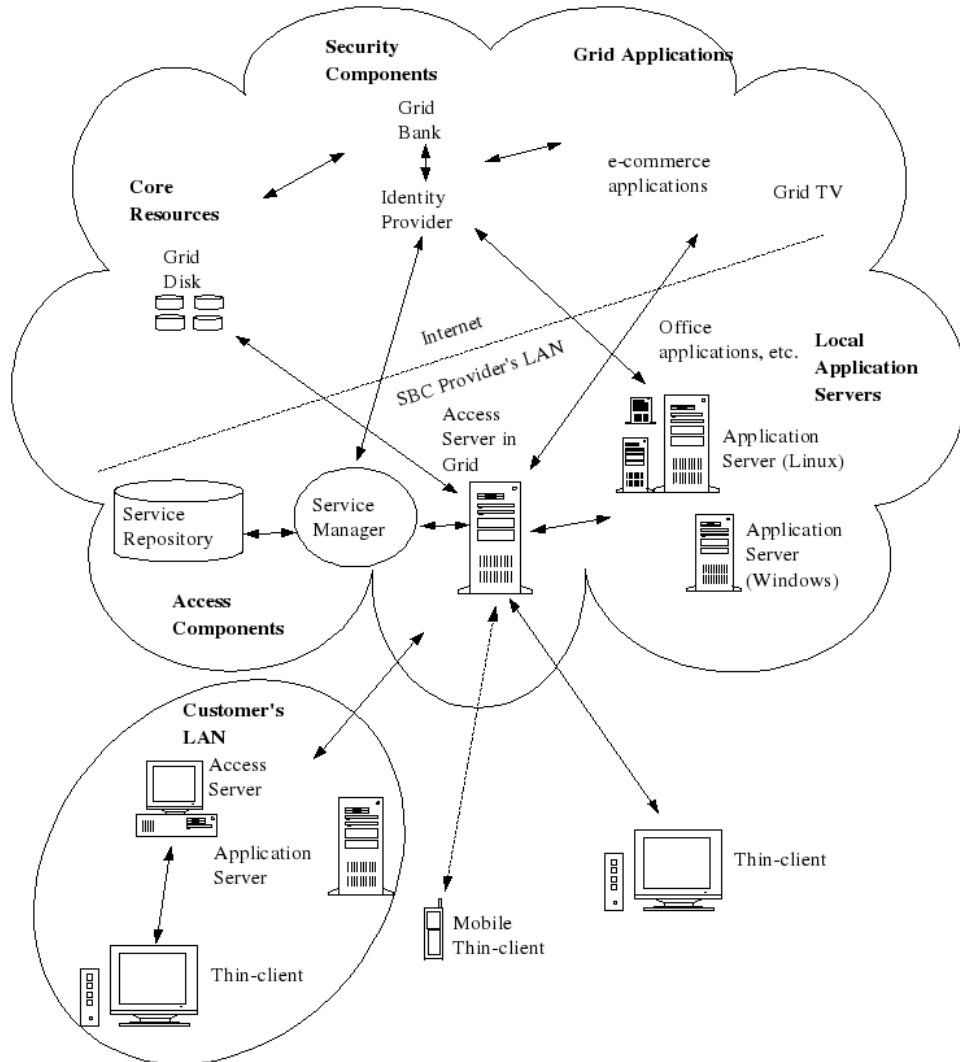


Figure 7.1: Future vision of server-based computing

where different resources both local and remote are accessible through one point of access and from several locations and devices. These resources can be connected to each other in a many ways but the client only has to know how to communicate

with the access point.

The applicability of SBC has earlier been restricted by weak network infrastructure and expensiveness of server hardware. Today networks are fast and diverse and processing power inexpensive. Now the facilities for SBC are there and it is gaining momentum.

# Appendix A

## Remote management system setup manual

Here is a short description on how to set up different parts of the remote management system. It contains the instructions for setting up both the client side and the server side tools. The instructions are meant for a linux administrator and might not be suitable for an unexperienced user.

1. Repository
2. Automatic installation
3. Virtual machine images
4. Virtual machine Debian packages
5. packagemanager.deb
6. dom0config.deb

### 1. Repository

Install the following packets. We used Ubuntu Feisty distribution.  
Required packages: apache2, gnupg, reprepro

The repository is basically a collection files that are accessible via file server. It can work on top of a ftp server or a HTTP server. In our system we have used Apache HTTP server. The repository is build into a subfolder of the web server's root. The configuration file of the repository is by default in conf directory

(`/var/www/yourdir/conf`) and is called `distributions.conf`. Below is our example of it.

### **distributions.conf:**

```
Origin: Hiptek repository Label: Hiptek repository Suite: feisty Codename: hiptek Version: 3.0 Architectures:
i386 amd64 ia64 Components: main Description: Hiptek repository for netgate purposes SignWith: yes
```

After configuring the repository we will make the signing keys for the repository. For that we use GnuPG (GNU Privacy Guard). Keys will be generated into the home folder of the user if not set otherwise.

### **gpg --gen-key**

This command creates both private and public key. The private key is used by Reprepro to sign packages and the public key by apt-tools on the client side.

Adding, removing and updating packages in the repository is done with Reprepro.

```
reprepro includedeb hiptek package.deb - adds and updates package.deb
reprepro remove hiptek package.deb - removes package.deb
```

## **2. Automatic installation**

Automatic installation is done with the Debian installer. Remote machine is started with a media containing a generic kernel and the installation software. We used both usb-stick and cd-rom. Usb-stick is created using syslinux and the cd-rom using isolinux. Usb-stick creation tools and St.Petersburg's special master boot record (MBR) that can be found from the repository. They are contained inside `bootingtools.tar.gz`, which can be found from the root of the repository, and inside `dom0config` package.

Installation tool is configured to boot with predefined parameters that lessen the involvement from the installer. Following example is from the `syslinux.cfg` used in our installation media.

Kernel boot parameters:

```
kernel linux append
preseed/url=http://wiki.hip.fi/ubuntu/feisty--ask-disk--preseed.txt locale=en-US
bootkbd=se console-setup/layoutcode=se console-setup/variantcode=nodeadkeys
netcfg/get_hostname=ubuntu-xen-server auto=true priority=critical
base-installer/kernel/linux/extra-packages-2.6=
pkgsel/install-pattern=~t^ubuntu-standard$ pkgsel/language-pack-patterns=
pkgsel/install-language-support=false acpi=off noapic vga=normal
initrd=initrd.gz ramdisk_size=14332 root=/dev/rd/0 rw -
```

This fetches a preseed file from our server. The preseed tries to answer to as many questions as possible so that user would not have to. Preseed files are very configurable and allow scripting. Preseed defines for example how the hardware

should be configured. It tells the installation from where it should download all the necessary installation medias. There you can also define your own repository that contains your virtual machines. You must also define location for the repository's public key, so that the installation can fetch it. If you need to install some extra software on top of the base installation you can set it in the preseeds. We have included the following: ssh, openvpn, dom0config, ubuntu-xen-server, rsync, packagemanager, mtools and syslinux. Dom0config and packagemanager are home made packages. Dom0config sets up the default network configurations and packagemanager starts the automatic updates. These two packages are described in more detail later.

### Preseed:

```
##### Localization #####
d-i debian-installer/locale string en_US
d-i console-keymaps-at/keymap select se

#####
##### Network config #####

d-i netcfg/choose_interface select auto
d-i netcfg/use_dhcp boolean true
d-i netcfg/get_hostname string no-dhcp-hostname
d-i netcfg/get_domain string no-dhcp-domain
d-i netcfg/wireless_wep string

#####
##### Mirror site settings #####

d-i mirror/country string FI
d-i mirror/http/hostname string ftp://ftp.funet.fi/pub/mirrors/archive.ubuntu.com/
d-i mirror/http/directory string /
d-i mirror/suite select feisty
d-i mirror/http/proxy string

#####
##### Partitioning #####

d-i partman-auto/method string regular
d-i partman-auto/choose_recipe select All files in one partition (recommended for new users)
d-i partman/confirm_write_new_label boolean true
d-i partman/choose_partition select Finish partitioning and write changes to disk
d-i partman/confirm boolean true

#####
##### Boot loader installation.

d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean true
d-i grub-installer/bootdev string (hd0,0)

#####
##### Package selection #####

tasksel tasksel/first multiselect standard
d-i pkgsel/include string ssh openvpn dom0config ubuntu-xen-server
rsync packagemanager mtools syslinux
```



```
#####
#### Finishing up the first stage install.

# Avoid that last message about the install being complete.
d-i prebaseconfig/reboot_in_progress note

#### Clock ####

d-i clock-setup/utc boolean true
d-i time/zone string Europe/Helsinki

#####
#### Apt Setup ####

d-i apt-setup/non-free boolean true
d-i apt-setup/contrib boolean true

# Additional repositories , local[0-9] available
d-i apt-setup/local0/repository string http://wiki.hip.fi/ubuntu hiptek main
d-i apt-setup/local0/comment string Hiptek virtual machines
d-i apt-setup/local0/key string http://wiki.hip.fi/ubuntu/aptpubkey
d-i debian-installer/allow_unauthenticated string true

#####
#### Account setup ####

d-i passwd/root-login boolean true
d-i passwd/make-user boolean true
d-i passwd/root-password password qwert0
d-i passwd/root-password-again password qwert0

#passwd          passwd/make-user          boolean true
d-i passwd/user-fullname string Admin account
d-i passwd/username string maintainer
d-i passwd/user-password password insecure
d-i passwd/user-password-again password insecure

#####
#### end scripts ####

# This command is run just before the install finishes , but when there is
# still a usable /target directory.
d-i preseed/late_command string wget http://wiki.hip.fi/ubuntu/feisty-late-preseed-cmd
    -O /target/root/late-preseed-cmd; chmod +x /target/root/late-preseed-cmd;
    /target/root/late-preseed-cmd;echo "late preseed command run" > /target/root/hellomessage

# This command is run after base-config is done, just before the login:
# prompt. This is a good way to install a set of packages you want, or to
# tweak the configuration of the system.
d-i base-config/late_command string wget http://wiki.hip.fi/ubuntu/feisty-late-base-config-cmd
    -O /root/late-base-config-cmd; chmod +x /root/late-base-config-cmd;
    /root/late-base-config-cmd;echo "late base-config command run" >> /root/hellomessage
```

### 3. Virtual machine images

To launch a virtual machine you need to have a virtual machine image, Xen compatible kernel/initrd and virtual machine configuration file.

There are several ways of creating virtual machines images for Xen and two different types of images. Images used with hardware virtualization are different

from the normal filesystem images. In our system we have only used normal filesystem images as they work with larger machine base.

The two ways we have used are 1.) Xen-tools 2.) Debootstrap by hand or copy existing root to a image. Xen-tools actually do all that is done with the second way with one command. Before running that command one should first configure the `/etc/xen-tools/xen-tools.conf` so that your default values will be of your design. Commandline parameters override the default values set in the `xen-tools.conf`.

`xen-tools.conf`:

```
dir = /opt/xen
debootstrap = 1
size = 5Gb      # Disk image size.
memory = 256Mb  # Memory size
swap = 512Mb    # Swap size
fs = ext3       # use the EXT3 filesystem for the disk image.
dist = feisty   # Default distribution to install.
image = full    # Specify sparse vs. full disk images.
gateway = 192.168.147.0
netmask = 255.255.255.0
passwd = 1
kernel = /boot/vmlinuz
initrd = /boot/initrd.img
mirror = ftp://mirror.switch.ch/mirror/ubuntu/
```

**`xen-create-image -ip ip -debootstrap hostname`**

`Ip` can be static or dhcp and `debootstrap` can be replaced with `rpmstrap` depending on your distribution.

The following example does the same as the `xen-create-image` except that one must afterwards make the xen configuration file for the virtual machine.

1. `dd if=/dev/zero of=empty.img bs=1024k count=1000`
2. `mkfs.ext3 empty.img`
3. `mkdir mnt ; mount -o loop empty.img mnt/`
4. `debootstrap -arch i386 feisty mnt/ ftp://mirror.switch.ch/mirror/ubuntu/`
5. `cp -dpR /lib*/modules/ mnt/lib*/modules/`
6. `umount mnt/`
7. `rmdir mnt`

What this does is it creates an empty image file, initializes it with ext3 filesystem and debootstraps feisty distribution into it.

Virtual machine specific configuration file:

```

kernel = '/boot/vmlinuz'
ramdisk = '/boot/initrd.img'
memory = '1024'
root = '/dev/sda1 ro'
disk = [ 'file:/opt/xen/domains/askoM-CH-1.3.0/disk.img,sda1,w',
         'file:/opt/xen/domains/askoM-CH-1.3.0/swap.img,sda2,w' ]
name = 'askoM'
vif = [ '' ]
on_poweroff = 'destroy'
on_reboot = 'restart'

```

The default location for virtual machine configuration files is `/etc/xen` directory. After you have created the virtual machine and the configuration file you can start the virtual machine with “`xm create`” command.

**xm create <configuration file>**

## 4. Virtual machine Debian packages

Debian packages can be made with different tools but they are all based on `dpkg`, which is the base of Debian package management system. Every file and folder in the building directory of the package are packaged into one file and when installed they are placed accordingly to the root of the client’s filesystem. A file `X` in folder `Y` under building folder will be placed into `/X/Y` in the client machine’s root. So the files should be placed so that the virtual machine configuration file is under `/etc/xen` and the image under `/opt/xen/domains/iname of virtual machine`.

As described earlier the virtual machine requires a working kernel and `initrd.img`, virtual machine image file and a configuration file to start. In our system we use the same kernel and `initrd` with every virtual machine. This makes the updating and configuration more simple.

The less data there is to transfer the faster the virtual machines can be installed. This is why we try to make as much at the client machine as possible. We use destination machines modules and kernel and we also make the swap file at the destination. This is done with scripts that are part of the debian package.

In addition to the virtual machine specific files there is also a control file directory in the building directory root named `DEBIAN`. This folder is required by the `dpkg`. In this folder we put configurations, version information and scripts. In our packages we have included `changelog`, `control`, `copyright` configuration files where the most important being the `control` file. We have also used two scripts, `postinst` script is used to set up the virtual machine environment and `prepm` script to clean up afterward.

**postinst:**

```

#!/bin/sh -e

xm='/usr/bin/which xm'
xen_conf=/etc/xen
vmType=askoM-CH
version=1.3.0
vm=$vmType-$version
image_root=/opt/xen/domains

# Phases:
# 1.) Create required directories
# 2.) Extract images and configurations
# 3.) Copy files to their locations
#
# Phases 1,2,3 done automatically
#
# 4.) Copy files to the image
# 5.) Make swap space
# 6.) Start the virtual machine
#

clear()
{
    /bin/umount /tmp/$vm
    /bin/rmdir /tmp/$vm
}

checkFiles()
{
    echo "Checking virtual machine files"
    echo " $image_root , $image_root/$vm/disk.img, /etc/xen/$vm.cfg"
    if [ -d $image_root -a -f $image_root/$vm/disk.img -a -f /etc/xen/$vm.cfg ]
    then echo "OK"
    else
        echo "False. Not all required files were found"
        exit 0
    fi
}

# 4. Append system modules to virtual machines image
appendModules()
{
    echo "Adding modules of the running system to virtual machines image"
    echo "running kernel" 'uname -r'
    if (/bin/mkdir /tmp/$vm)
    then continue
    else
        echo "unable to make directory /tmp/$vm!"
        exit 0
    fi
    if (/bin/mount -o loop $image_root/$vm/disk.img /tmp/$vm)
    then continue
    else
        echo "unable to mount $image_root/$vm/disk.img to /tmp/$vm!"
        /bin/umount /tmp/$vm
        exit 0;
    fi
    if (/bin/mkdir /tmp/$vm/lib/modules/'uname -r')
    then continue
    else
        echo "unable to create directory /tmp/$vm/lib/modules/" 'uname -r' "!"
        clear
        exit 0
    fi
    if (cp -r /lib/modules/'uname -r' /tmp/$vm/lib/modules/)

```

```

        then continue
    else
        echo "Unable to copy modules!"
        clear
        exit 0
    fi
    if (/bin/umount /tmp/$vm)
        then /bin/rmdir /tmp/$vm
    else
        echo "unable to umount temporary directory"
        exit 0
    fi
    echo "OK"
}
# 5. Make swap space
makeSwap()
{
    echo "Creating swap space for $vm"

    dd if=/dev/zero of=$image_root/$vm/swap.img bs=1M count=500

    if(mkswap $image_root/$vm/swap.img)
        then echo "ok!"
    else
        echo "failed!"
    fi
}
# 6. Start virtual machine
startVM()
{
    echo "Starting virtual machine..."

    if($xm create $vm.cfg)
        then echo "ok!"
    else
        echo "failed!"
        echo "You have xend installed and running, right?"
    fi
}
#####
#MAIN#
#####
checkFiles
appendModules
makeSwap
startVM

```

**prerm:**

```

#!/bin/sh -e
# 1.) Stop the running virtual machine
# 2.) Delete the swap image
vmType=askoM
version=1.3.0
serverName=askoM-CH
vm=$vmType-$version
imageRoot=/opt/xen/domains

# Stop running virtual machine
stopVM()
{
    echo "Stopping virtual machine $serverName"
    if( xm shutdown $serverName )

```

```

        then echo "OK!"
    else
        echo "Failed! Maybe it was not running?"
    fi
}
# Delete the swap image
delSwap()
{
    echo "Deleting virtual machines ($vm) swap image"
    if ( rm $imageRoot/$vm/swap.img )
        then echo "OK!"
    else
        echo "Failed! Check if it was left behind (/opt/xen/domains/$vm)."
    fi
}
stopVM
delSwap

```

**control:**

```

Section: devel
Priority: optional
Installed-Size: 3300
Maintainer: Jukka Kommeri <kommeri@cern.ch>
Standards-Version: 3.7.2.1
Package: askom-ch
Version: 1.3.0
Architecture: i386
Description: Light application server VM image with xfce4 desktop

```

After you have all the files in place, the creation of the Debian package is done with the following command.

**dpkg -b directory packagename.deb**

**5. packagemanager.deb**

The package manager fetches periodically packagelists from a server and updates machine's package collection accordingly. For this, we have set up a Rsync server which serves the packagelists. The server has a client specific account which is only accessible with SSH+Rsync. The server side is protected with rssh and client machines can only run rsync on the server. For it to work automatically, we have included the servers public key to the package manager package.

```

#!/bin/sh

sleep 3
rsync -az -e "ssh -i /root/.ssh/netgate -l netgate"
    netgate@wiki.hip.fi:/opt/rsync/test/ /etc/pkgsync/

if [ -s /etc/pkgsync/musthave ]
    then echo "Checking packalists for updates"
        pkgsync
    else echo "The file /etc/pkgsync/musthave file size is zero. It cannot be!"
fi

```

The script that fetches the packagelists is placed in the `/etc/cron.daily` so that by default it would be run around 4 am.

Server side has two configuration files. One for Rsync and one for Rssh. In `rsyncd.conf` we define the modules that are shared. The modules must contain at least the `musthave` file, which defines the packages of the client machine. `Rssh.conf` is configured to restrict the use of the shell just to `rsync` and set the file permissions.

### **`/etc/rsyncd.conf:`**

```
uid = root
gid = root

max connections = 10
#motd file = /etc/rsyncd/rsyncd.motd

[packetlist1]
    /opt/rsync/test/
    read only = yes
```

### **`/etc/rssh.conf:`**

```
logfacility = LOG_USER
allowrsync
umask = 022
user=netgate:011:10001: # rsync, with no chroot
```

## **6. Dom0config.deb**

The `dom0config` package configures the `xen dom0` which is the privileged `xen` domain. It sets up the network and adds our own network configurations to the `Xen` daemon configurations. `/etc/xen/xend.conf` is edited so that it uses `network-dummy` script. Instead of running the default network script we use our own network configurations.

### **`iptables.conf:`**

```
:PREROUTING ACCEPT [22:2922]
:POSTROUTING ACCEPT [2:156]
:OUTPUT ACCEPT [1:72]
-A PREROUTING -i xenbr0 -p udp -m udp --dport 53 -j DNAT --to-destination 153.1.63.37:53
-A POSTROUTING -s 192.168.147.0/255.255.255.0 -o eth0 -j MASQUERADE
COMMIT
*filter
:INPUT DROP [20:2762]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [543:80392]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -i xenbr0 -j ACCEPT
-A INPUT -s 192.168.147.0/255.255.255.0 -p icmp -j ACCEPT
-A INPUT -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 123 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
```

```

-A INPUT -p udp -m udp --dport 1194 -j ACCEPT
-A INPUT -j LOG
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 20:21 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 22 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 53 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 80 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 110 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 119 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 123 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 143 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 443 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 443 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 465 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 514 -j ACCEPT
-A FORWARD -s 193.168.147.0/255.255.255.0 -p tcp -m tcp --dport 631 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 636 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 902 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 993 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 995 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 1194 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 3389 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 3690 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 8001 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 6666:6669 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 6000:6009 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 9100 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p tcp -m tcp --dport 11371 -j ACCEPT
-A FORWARD -s 192.168.147.0/255.255.255.0 -p udp -m udp --dport 45000 -j ACCEPT
-A FORWARD -p udp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 69 -j ACCEPT
-A FORWARD -p udp -s 192.168.147.0/24 -d 192.168.147.0/24 --sport 69 -j ACCEPT
-A FORWARD -p udp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 32765:32768 -j ACCEPT
-A FORWARD -p tcp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 32765:32768 -j ACCEPT
-A FORWARD -p udp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 2049 -j ACCEPT
-A FORWARD -p tcp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 2049 -j ACCEPT
-A FORWARD -p udp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 111 -j ACCEPT
-A FORWARD -p tcp -s 192.168.147.0/24 -d 192.168.147.0/24 --dport 111 -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 22 -j ACCEPT
-A FORWARD -p icmp -j ACCEPT
-A FORWARD -p udp --sport 68 --dport 67 -j ACCEPT
-A FORWARD -p udp --sport 67 --dport 68 -j ACCEPT
-A FORWARD -i xenbr0 -j ACCEPT
-A FORWARD -o xenbr0 -j ACCEPT
-A FORWARD -j LOG
COMMIT

```



# Appendix B

## CERN library thin client system

1. Introduction
2. Network
3. Servers
4. Thin clients
5. Customization
6. Software

### **1.Introduction**

The thin client installation for CERN library operates with one server machine that handles multiple thin clients. This server uses virtualization to ease the management of updates and to increase overall security. The server hardware runs three servers. One is the virtualization platform, the Xen dom0, and its name is pcdsusi02.cern.ch. The two other machines are virtual and they are called askoM and ltsp42.

All the machines both real and virtual get their software from Ubuntu Feisty distribution. Thin clients can be placed around the library and connected to the server using wireless bridges. These brigdes give more freedom to the placement of the thin clients and remove the need for a network infrastructure.

### **2. Network**

Pcdsusi02, the dom0, is the only machine visible to the CERN network. It is the gateway and the firewall for the virtual machines. It uses CERN DHCP to set its ip address. Pcdsusi02 is registered to the DNS of CERN. Virtual machines askoM, ltsp42 and thin clients are in a private local area network and use pcdsusi02 to

access outside network. Network configuration of the system is illustrated in Figure B.1. Virtual LAN in the figure illustrates the network inside pcdsusi02.

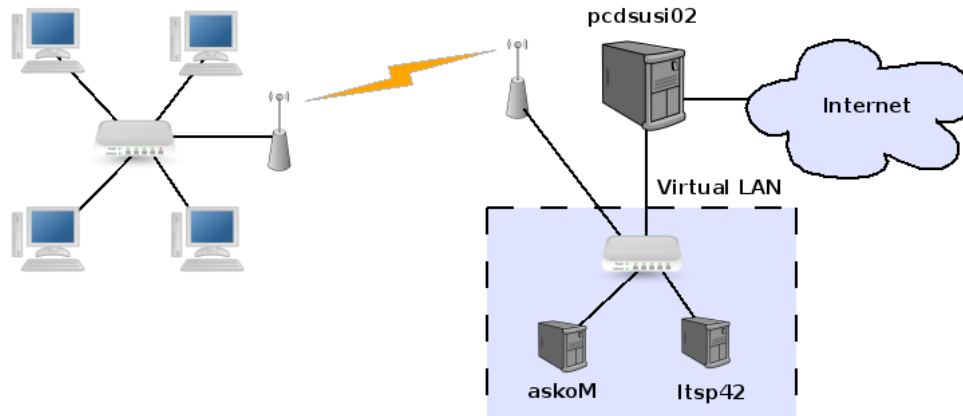


Figure B.1: The network of thin client system

To make the placement of thin clients more flexible, we have also added wireless bridges to the private local area network. These bridges connect thin client clusters to the servers. The bridges have their own IP addresses, which are 192.168.147.2, 192.168.147.3 and 192.168.147.4. Address 192.168.147.2 belongs to the root bridge that has a wire connection to the server. The other bridges connect only to the root bridge.

The address space of thin clients in the private LAN is limited to 192.168.147.100-192.168.147.120 meaning that there can be max 21 thin clients behind one server. This is also a reasonable limit of thin clients per one server.

### 3. Servers

Pcdsusi02.cern.ch is the host for the virtual servers. It uses Xen paravirtualization to multiplex the hardware. The server is installed using a remote installation system. This system installs basic Ubuntu Feisty with Xen. It also adds a preconfigured firewall and launches a automatic management system, that keeps the software and virtual machines up to date.

Virtual servers:

- **ltsp42**-192.168.147.11, is a Ubuntu Feisty with ltsp4.2 tools.
- **askoM**-192.168.147.12, is a Ubuntu Feisty with XFCE4 desktop environment and all the necessary office tools. This is the actual server that the users are allowed to use and where the guest user accounts and home folders reside.

### 4. Thin clients

Thin clients boot from the access server, ltsp42, and get all the needed software from there via pxe/tftp and nfs. Once initialized the thin client connects to askoM where the actual user session is run and all the user software is maintained and executed. Connection between the thin client and server is maintained inside a ssh-tunnel, which secures the the client session as it has to go through a less secure wireless connection.

## 5. Customization

askoM has an admin user account. The personalized settings of this account are used to initialize the home folders of the guest accounts used by the thin clients. Guest users' home folders are reinitialized once a day at 4 am. Administration can also force this operation with root account. The command for that is `/root/remakeHomeFolders.sh`. After running this command the thin clients should be booted. In addition the server will remake all the guest accounts when rebooted and wipe out all related files.

To use the admin user account one must log in with the Xephyr program. This program can be used to make remote x-connections to even lxplus. In our case it is used to connect to localhost "Xephyr :3 -query localhost" and log in with the "guest" account. When logged in one can make changes to the desktop configurations. These changes will be applied to other guest accounts when the home folders are reinitialized.

## 6. Software

The thin client system uses only open source software. All the software is installed from the packages of the Ubuntu Feisty distribution. There are three main software assemblies: Xen virtualization software, LTSP thin client management software and XFCE4 desktop environment. In addition there are some scripts that automate the management of services in different servers.

Xen virtualization tools include the Xen hypervisor and its management tools. These tools make it possible to create and destroy virtual machines. They also include rich set of other management tools to for example. backup virtual machines or migrate them to other machines. LTSP software include several server side components such as dhcp,tftpd,nfs server and ltsp's own scripts for managing heterogenous hardware pool. These services make it possible to boot thin clients from the network.

XFCE4 is a light desktop environment for Linux machines. It is easily configurable to meet different needs. In our case we have tried to configure the environment to meet the basic library use. The menu of the desktop contain software organized into intuitive categories. Software collection is limited to the basic office tools which include Open Office, Firefox with flash, Acrobat reader and some basic tools provided by the XFCE4 by default. In addition there are

some software for remote sessions such as rdesktop, ssh-client and Xephyr. Most important software have a shortcut on the panel of the desktop.

# Bibliography

- [1] MyPLC User's Guide.  
<http://www.planet-lab.org/doc/myplc>.  
Referenced 2.12.2007.
- [2] Xen basics.  
<http://www.nodemaster.de/24-0-xen-basics.html>.  
Referenced 22.11.2007.
- [3] *Understanding the Remote Desktop Protocol (RDP)*, March 2007. Article ID : 186607, Revision : 2.2.
- [4] Environmental comparison of the relevance of pc and thin client desktop equipment for the climate, 2008.
- [5] Netvoyager plc . Server-based computing explained.  
<http://www.netvoyager.co.uk/general/sbce.html>.  
Referenced 21.8.2006, Last edited.
- [6] Sun Microsystems . Sun ray server software 3.1 administrator's guide for the linux operating system.  
<http://docs.sun.com/app/docs/doc/819-2389>.  
Referenced 26.09.2006, Last edited 9.12.2005.
- [7] VMWare . Technology preview for transparent paravirtualization.  
<http://www.vmware.com/interfaces/techpreview.html>.  
Referenced 16.5.2008.
- [8] K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2006. ACM.
- [9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

- [10] D. Blackman. Debian package management, part 1: A user's guide. *Linux J.*, 2000(80es):12, 2000.
- [11] P. Chaganti. *Xen Virtualization, A practical handbook*. Pact Publishing Ltd., 2007.
- [12] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.
- [13] C. L. Coffing. An x86 protected mode virtual machine monitor for the mit exokernel.  
Referenced 21.1.2007.
- [14] R. J. Creasy. The origin of the VM370 time-sharing system. *IBM Journal of Research & Development*, September 1981.
- [15] B. des Ligneris. Virtualization of linux-based computers: The linux-vserver project. In *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*, pages 340–346, Washington, DC, USA, 2005. IEEE Computer Society.
- [16] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, and P. Toft. Smartfrog: Configuration and automatic ignition of distributed applications. Technical report, HP, 2003.
- [17] S. Greenberg. What is thin client computing. *For the Record*, July 2000.
- [18] X. Grehant, O. Pernet, S. Jarp, I. Demeure, and P. Toft. Xen management with smartfrog.
- [19] B. Havard. HPC Virtualization with Xen on Itanium. Master's thesis, Norwegian University of Science and Technology, July 2005.
- [20] HP Development Company .  
hewlett-packard server-based computing - solution overview.  
<http://activeanswers.compaq.com/ActiveAnswers/cache/70284-0-0-0-121.html>.  
Referenced 26.09.2006, Last edited 21.8.2006.
- [21] A. Isotton. Debian repository howto.  
<http://www.debian.org/doc/manuals/repository-howto/repository-howto>.
- [22] M. T. Jones. Discover the linux kernel virtual machine.  
<http://www.ibm.com/developerworks/linux/library/1-linux-kvm/>,  
April 2007.

- [23] J. Kanter. *Understanding Thin-Client/Server Computing*. Microsoft Press, 1998.
- [24] K. V. Kaplinsky. Vnc tight encoder - data compression for vnc. In *Proceedings of the 7th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists Modern Techniques and Technology MTT 2001*. IEEE Standards, 2001.
- [25] A. Kros and M. A. Margevicius. Thin-client shipments stage strongest growth since 2000. Technical report, Gartner, June 2006.
- [26] A. Lai and J. Nieh. Limits of wide-area thin-client computing. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 228–239, New York, NY, USA, 2002. ACM.
- [27] E. Leader. Thin clients trump pcs on energy consumption. <http://www.environmentalleader.com/2008/03/26/thin-clients-trump-pcs-on-energy-consumption/>, 2008.
- [28] J. Liu, W. Huang, B. Abali, and D. K. Panda. High performance vmm-bypass i/o in virtual machines. In *USENIX-ATC'06: Proceedings of the Annual Technical Conference on USENIX'06 Annual Technical Conference*, pages 3–3, Berkeley, CA, USA, 2006. USENIX Association.
- [29] J. N. Matthews, E. M. Dow, T. Deshane, W. Hu, J. Bongio, P. F. Wilbur, and B. Johnson. *Running Xen: A Hands-On Guide to the Art of Virtualization*. Prentice Hall, April 2008.
- [30] J. McQuillan. LTSP - Linux Terminal Server Project - v4.1. <http://ltsp.mirrors.tds.net/pub/ltsp/docs/ltsp-4.1-en.html>. Revision 4.1.3-en.
- [31] Microsoft. *Remote Desktop Protocol (RDP) Features and Performance*. [http://www.thinclient.net/technology/RDP\\_Features\\_and\\_Performance.htm](http://www.thinclient.net/technology/RDP_Features_and_Performance.htm).
- [32] Microsoft. *Configuring authentication and encryption*, January 2005. <http://technet2.microsoft.com/windowsserver/en/library/a92d8eb9-f53d-4e86-ac9b-29fd6146977b1033.mspx?mfr=true>.
- [33] T. Niemi, M. Tuisku, and M. Kokkonen. A server-based computing system based on open source software. referenced 28.9.2006, August 2006.
- [34] A. Nye. *X Protocol reference Manual for X11 Version 4, Release 6*. O'Reilly & Associates, inc, 1995.
- [35] Qumranet. KVM - Kernel-based Virtualization Machine. [http://www.qumranet.com/files/white\\_papers/KVM\\_Whitepaper.pdf](http://www.qumranet.com/files/white_papers/KVM_Whitepaper.pdf).

- [36] S. Regis. Getting started with NX .  
<http://www.nomachine.com/documents/getting-started.php>.  
 Referenced 7.5.2008, Last edited 1.8.2007.
- [37] M. Revett, I. Boyd, and C. Stephens. Network computing: a tutorial review.  
*Electronics & Communications engineering journal*, February 2001.
- [38] T. Richardson. The rfb protocol. Technical report, RealVNC Ltd, 2007.
- [39] R. Rodstein. Chapter 2: The citrix access gateway product line.  
<http://seoutsourcing.com/node/15>.  
 Referenced 2.10.2008.
- [40] R. Rose. Survey of system virtualization techniques. cite-seer.ist.psu.edu/rose04survey.html, 2004.
- [41] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.
- [42] E. Salmio, R. Kivi, and M. Suvanén. Information society structures in educational institutions - results of the surveys 2004 and summary of the years 2004. Technical report, Ministry of Education, 2004.
- [43] R. W. Scheifler and J. Gettys. The x window system. *ACM Trans. Graph.*, 5(2):79–109, 1986.
- [44] R. S. Shuford. DEC Terminals. <http://www.cs.utk.edu/shuford/terminal/dec.html>.  
 Referenced 1.10.2008.
- [45] J. Sugerman, G. Venkitachalam, and B. hong Lim. Virtualizing i/o devices on vmware workstation’s hosted virtual machine monitor. In *Proceedings of the 2001 Annual Technical Conference*. USENIX Association, 2001.
- [46] A. Tijms. Binary translation: Classification of emulators. Technical report, Leiden Institute of Advanced Computer Science, 2000.
- [47] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. M. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.
- [48] J. Wack, K. Cutler, and J. Pole. Guidelines on firewalls and firewall policy. Technical report, National Institute of Standards and Technology, 2002.
- [49] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble. Rethinking the design of virtual machine monitors. *Computer*, 38(5):57–62, 2005.
- [50] A. Whitaker and S. D. Gribble. propos of machine virtualization.  
[http://denali.cs.washington.edu/pubs/distpubs/slides/retreat\\_july\\_2001/VM\\_retreat\\_2.pdf](http://denali.cs.washington.edu/pubs/distpubs/slides/retreat_july_2001/VM_retreat_2.pdf).



- [51] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications, 2002.
- [52] S. J. Yang, J. Nieh, M. Selsky, and N. Tiwari. The performance of remote display mechanisms for thin-client computing. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 131–146, Berkeley, CA, USA, 2002. USENIX Association.
- [53] A. Zeichick. Processor-Based Virtualization, AMD64 Style, Part II. [http://developer.amd.com/article\\\_print.jsp?id=15](http://developer.amd.com/article\_print.jsp?id=15). Referenced 3.11.2007, Last edited 30.6.2006.