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

Author:	Jukka Kommeri
Name of the thesis:	System management in Server Based Computing with virtualization
Date:	24 April, 2007 **Number of pages:** 50
Department:	Electrical and Communications engineering
Professorship:	S-38
Supervisor:	Professor Jörg Ott
Instructor:	Tapio Niemi, PhD, Helsinki Institute of Physics

The purpose of this thesis is to research 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 will makes installation of remote machines more 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 to several locations and many times. In a bigger organization this can build up to be a big expense. On the other hand the smaller organizations cannot always afford a real administrator and the administrating work falls to the person most capable. This results to insecure and unreliable installations. For these companies it would be beneficial to be able to have the services ready to use without any expertise.

We solved this by providing a system for distibuting centrally managed services. This was done by enclosing generic services and 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 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 remote machines needs. The system provides the means for automated deployment of the base operating system and virtualization tools over Internet and 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 the 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

TEKNILLINEN KORKEAKOULUDIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Jukka Kommeri
Työn nimi:	Palvelun hallinta palvelin keskeisessä ympäristössä
Päivämäärä:	24. huhtikuuta 2007 **Sivuja:** 50
Osasto:	Sähkö ja tietoliikennetekniikan osasto
Professuuri:	S-38
Työn valvoja:	Professori Jörg Ott
Työn ohjaaja:	Tapio Niemi, FT, Helsinki Institute of Physics
Tiivistelmä...	
Avainsanat: sbc, virtualization, xen, ltsp	

Acknowledgements

Some text...

CERN Geneva, 24 April, 2007

Jukka Kommeri

Table of Contents

List of Figures

List of Tables

Abbreviations

APT	Advanced Packaging Tool
CERN	l'Organisation Européenne pour la Recherche Nucléaire
FS	File System
GDI	Graphics Device Interface
HIP	Helsinki Instute of Physics
HTTP	Hypertext Tranfer Protocol
HTTPS	Hypertext Tranfer Protocol Secure
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

Chapter 1

Introduction

Server Based Computing (SBC) is a model in which all the applications are installed, managed and executed on a server [30]. Users' computers are considered as thin clients that exchange keyboard, mouse and screen information with the server. Depending on the implementation, no actual programs are executed on the client machine. Meaning that the balance of the execution load between the server and the client may vary.

SBC itself is nothing new as it has been around for many decades. It has been a good solution for many cases but not a complete solution due to its dependency on network speed and computing power. 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 the server. Adding new terminals requires no installation work because 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.

Interest on this topic has not only been academic. 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 some commercial SBC solutions such as the Sun

Microsystems' Sun Ray [26], HP's Server Based Computing solution [11] and a newcomer Panologic [1] as well as non commercial ones such as LTSP [2], K12LTSP [3]. According to Gartner's studies [21] the thin client market has increased 38% in the year 2006 which is according to them the highest since the year 2000.

The problem is that one LTSP server can only serve a limited number of users. In a big organization it is required that more than one server is administered. The bigger the organization is, the more work is required. Also, different services may introduce more hardware requirements. To make it easier to handle multiple servers and services there should be a central service distributor. A server for servers that could be used to set up the SBC system in geographically dispersed sites. Centrally maintained sets of services for different applications could be distributed to different locations. These services could also be replicated and the same environments could be set up in various locations.

This thesis is structured as follows: In Chapter two we will look into different SBC technologies and define what is SBC 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 6. Its properties will be further analyzed in the following evaluation chapter. Finally we we will summarize the thesis with the conclusion chapter.

[1] www.panologic.com
[2] www.ltsp.org
[3] www.K12LTSP.org

Chapter 2

Problem Definition

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

2.1 Netgate 2

Netgate 2 [1] is a research project funded by the Finnish National Technology Agency (TEKES) and Finnish industrial partners. The project is performed 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, Terminal Grid computing, and Grid business research. This thesis is done as a part that research and concentrates on the Terminal Grid computing i.e. Server Based Computing.

The Server Based Computing part of the project tries to find a way to simplify the installation and management of remote systems: Making it possible for a

[1] http://tek.hip.fi/opencms/opencms/projects/finnish/index.html#netgate-II

nontechnical person to set up complicated services on their premises and also relieving him from the administration. By doing this it will decrease overall administration costs and increase the quality of administration for the end systems i.e. the new installations would be more robust and secure.

2.2 Research Problem

The challenge of installing servers and services into them is that it requires a lot of knowledge to be performed. In many cases the administrator is not fully qualified to do the installation but is selected to the task since he has the best knowledge about the subject [15]. This can lead to unstable and insecure systems. To be sure that the system works the way it should it need to be installed and administered by professionals. Problem is that the usage of professionals like having your own administration is expensive.

To make the services available in working environments that lack the required skills a out of the box solution would be needed:

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

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

We solve this by using virtualization and automated installation of the client server. Services are placed inside virtual machine images and these virtual machine images are then delivered to the client machines. Virtualization makes it be possible to run different versions of the same system in parallel to test whether the new version works the way it should. These kinds of features are certainly important in the case critical services.

2.3 CERN Library Use-case

CERN is the world's largest particle physics laboratory [1]. It is situated on the border of France and Switzerland just next to Geneva. It employs nearly 3000 permanent 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.

Cern library provides, among other things, 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 CERN Windows domain. The installation time of one of these machines can take about an hour and requires involvement from the librarians. This procedure has to be repeated regularly otherwise the machines would slow down. Attending to computers is not part of the core knowledge of the librarians and it interferes with the normal responsibilities.

The library is a good place to pilot the product of this thesis. Replacement of CERN library's public terminals with a thin client system has many benefits. Thin clients are silent and easy to maintain. Installation of thin clients does not require any involvement because they need no 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 a bit more time but with our system it can also be done automatically. All the 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 relieves 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. New thin clients have a much longer life span than normal PCs, which in part reduces the administration work. Replacement of the desktop PCs with completely silent thin clients will also give the atmosphere of the library a well come silence.

[1] `http://public.web.cern.ch/public`

Chapter 3

Server Based Computing

Local area networks and wide area networks have made 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), now these can be located on central servers and made reachable by others.

Centralization is not only limited to storage services but it can also be used for execution of programs. Due to the development of the networks and the exponential growth of processing power, the servers can now serve increasing amount of client applications. The benefits of this transition from local execution to centralized is the decreased amount of administration work and increased system security.

SBC system resembles the old mainframe systems that were used in the early times of computers. The difference being now that the servers are a lot smaller and offer users more usable user interfaces [27].

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.
- Show graphics.
- Transmit information from its peripherals such as keyboard, mouse or USB device.

This kind of device is called a thin client[30].

The thin client is not a special device. Almost any PC can be used as a thin client. The requirements for thin clients are low since most or all of the real computation is done elsewhere. Thin client do not need hard drives, powerful processors or powerful graphics cards. Having less powerfull components means less energy consumption and less heat production.

The life cycle of the old and obsolete PCs can be extended while using as thin clients. The requirements for a thin client vary on how much of the software is executed on remote servers [14].

3.1 Existing Thin Client Middleware

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. Protocols balance between the load on the network and the load on the thin client: Usage of higher level graphics decreases the need to transmit data but means that the thin client has to compute high level graphic primitives to form the image. The usage of raw pixel encoding sends more information over network but requires less computation from the client. Meaning also that if the client does not need to support the higher level graphics it is easier to implement. Some protocols adjust to the network by sending updates less frequently and using caching and compression [41].

There has been a lot of development in the field of thin client protocols. Every new version brings performance improvements and makes the comparison of protocols more difficult. In the following sub-sections we will introduce a few popular protocols that are used in different scenarios.

3.1.1 ICA - Independent Computing Architecture

ICA [19] is a proprietary protocol owned by Citrix. Citrix is one of the SBC pioneers and offers multiple commercial solutions for both Windows and UNIX platforms. Its SBC is based on the independent computing architecture (ICA) though they also offer web-based solutions. ICA is a lightweight protocol that is

usable in low bandwidth networks such modem connections. It can be used to share both Windows and Unix based operating systems.

3.1.2 RDP - Remote Desktop Protocol

RDP [3] is Microsoft's proprietary protocol. RDP is an extension to ITU-T T.120 application sharing protocol family. It is used to connect to Windows Terminal Services.

RDP sessions can be configured to meet various needs. It supports different levels of encryption. The traffic between client and server can be secured with the transport layer security (TLS) [25] and compression is also supported. RDP uses high 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 [24].

RDP clients are made for Windows platforms. Though there is open source clients for X window systems such as Rdesktop [1] and RDP servers such as Xrdp [2].

3.1.3 VNC - Virtual Network Computing

VNC is an open source thin client protocol. It is based on Remote Frame Buffer (RFB). RFB is a thin client protocol that works on frame buffer level and this makes 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. [31].

VNC represents 2D graphics and raw graphics end of the protocols. Client retrieves pixel information from the server. Because of this clients are easier to implement and do not need a heavy system such as X Window to work. Due to this fact it can be used in many light devices such as mobile phones and personal digital assistants [41].

[1] www.rdesktop.org
[2] xrdp.sourceforge.net

Original VNC protocol did not come with compression or encryption support but they have been added to some derived products such as TightVNC [3]. Compression decreases the latecies of VNC in low bandwith networks[20].

3.1.4 X Window System

X Window System is the base for graphical user interfaces 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 machine or on a remote machine. Client and server use X protocol to communicate with each other [29].

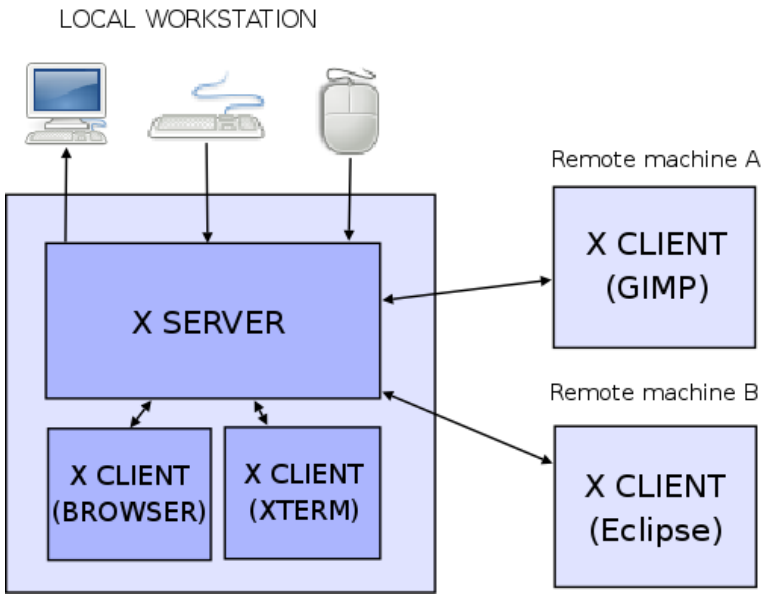

Figure 3.1: Simple X window system architecture

X window system is designed for local area networks (LAN). Though it uses high 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. That also makes it the best quality protocol [41].

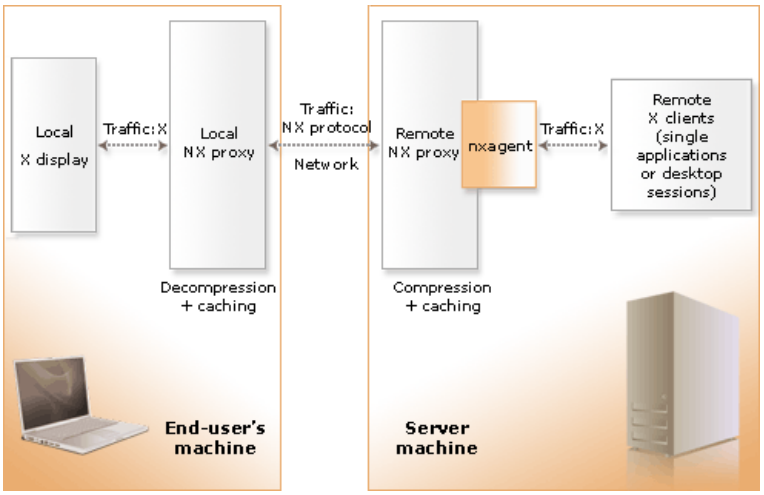

Figure 3.2: NX architecture (Courtesy of [28])

3.1.5 NX - New X

NX is a protocol developed to enhance the basic X window system. It acts as a buffer between X window system's client and server. It can store the session information on the remote machine making the X window system more resilient towards network errors. NX provides an encrypted and compressed link between client and server machine [28]. The protocol can also be adjusted to meet different network connections by suppressing the updates of the X window system. Figure 3.2 illustrates the architecture of the NX protocol.

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 and it has also been the base for the foundation of a new Linux distribution K12LTSP [4]. At the time of writing of this thesis LTSP is in version 5.0 with the codename MueKow[5].

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

[3] http://www.tightvnc.com
[4] www.k12ltsp.org
[5] wiki.ltsp.org/twiki/bin/view/Ltsp/MueKow

fat clients to a Linux server from different networks. Main usage of LTSP is to connect workstations in a lan to a central server, LTSP access server.

One does not need to install any operating systems to the workstations. They are booted with small Linux that is loaded either with PXE boot or with local medias such as floppy, CD-Rom or USB stick. No hard discs are used which means that you can use your current PC with an operating system and it will remain intact while using it as thin client. All required tools for managing thin client and graphical sessions are mounted from a network drive using NFS (Network File System). All the thin client does is run X server and connect to a session manager on the server in the LAN [23].

Chapter 4

Virtualization

In this chapter we will take a look at the world of virtualization. We will gather some information about what it is and why its usage would be beneficial.

In the world of computing, virtualization is a widely used term. There are many meanings to it and it is being applied in many forms. Basically it refers to the abstraction of various resources. Virtualization provides standard interfaces for applications and operating systems and removes their dependency on the underlying hardware. It can be used for example to multiplex hardware resources or to make them look like something else. One can virtualize complete platforms or just parts of it.

A good example of the virtualization is the Java programming language. Java code is compiled against the java environment and run on top of java virtual machine. The Java virtual machine always look 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 will concentrate on the 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 virtualization is used to decrease the proliferation of server machines and to improve their cost efficiency. Not only does it reduce the requirement for hardware but also reduces other physical costs such as electricity and space [33].

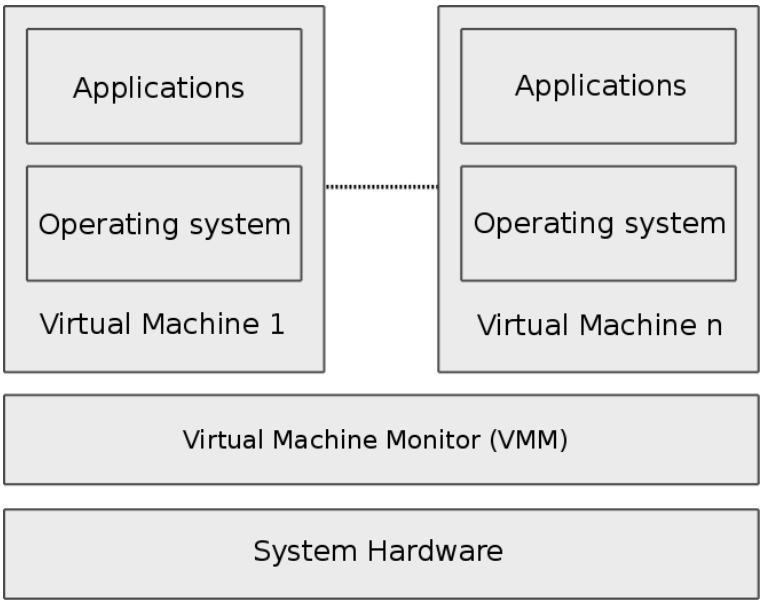

Figure 4.1: Virtualization architecture

With virtualization it is possible to divide the physical hardware among several virtual virtual machines (VM). These VMs are scheduled 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 would on a normal physical machine. VMs are isolated from each other by the VMM so that they cannot affect each other. As they would be when installed on their own machines [32].

VMM can reside either on top of an running operating system or directly on top of hardware. Running on the VMM on top of an OS introduces a lot a overhead but it is easy to set up for example testing environments.

4.2 Benefits

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. To have better fault tolerance, the applications should be distributed among several either physical or virtual machines. The need for fault tolerance and the inexpensiveness of 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 [33].

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 you have 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 on one service to gain administrative privileges on that machine. If there were other services running on the same machine, they would also be compromised. Placing applications into their own virtual machines creates a protective barrier between them [39].

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 to the new machines and then copy existing virtual machines on top of that. This cuts down the installation times and provides a way to run several versions of the same software in parallel. With virtualization you can set up testing, development and production versions of the system on the same machine.

4.3 Methodology

The x86 architecture is not virtualizable by design. This is due to the privilege levels of the instructions of the processor. The right to execute certain instructions depends on from which level it was executed [10]. Figure 4.3 illustrates the ring structure, the level 0 also known as kernel mode is able to execute all instructions and level 3, guest mode, a subset of this. Normally operating systems

execute at the level 0 and applications at level 3. In virtualized environment the virtual machine monitor works at the level 0 and guest operating systems on a higher level. Without any virtualization support the virtual machine running on the higher level would fail to execute any privileged instruction [35].

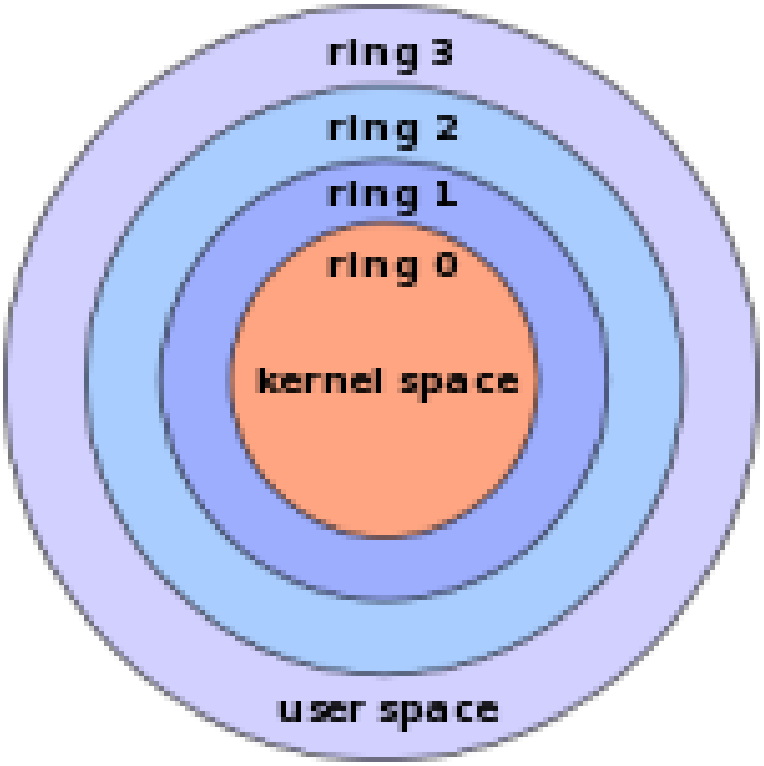

Figure 4.2: x86 ring model (Courtesy of [2])

There are many solutions to x86 architectures deficiencies. Here we introduce three main categories which are used in most of the current virtualization solutions: Paravirtualization, Software virtualization and Hardware virtualization

4.3.1 Paravirtualization

The architecture used with paravirtualization was originally developed by IBM and utilized in the VM operating system [1]. The term paravirtualization was introduced at 2001 [38] by Denali group [2] that uses paravirtualization in its OS.

[1] http://www.vm.ibm.com/
[2] http://denali.cs.washington.edu/

Since then paravirtualization has gained much popularity and is now used by leading virtualization developers such as Xensource and VMWare .

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. Usage of paravirtualization requires modifications to the guest operating systems. These modifications reduce virtual machine monitor's complexity and enhance its performance. The original machine instructions are either modified or excluded. Privileged commands are made to communicate with virtual machine monitor [39].

A problem with paravirtualization is that owners of proprietary operating systems might not be willing to modify their OS. Some OS instructions need to be modified for the virtual machine to able to operate on a higher privilege level [6]. The patching of any existing operating systems requires a lot of knowledge and work. This complicates the selection of the OS flavour of your choice and makes you want to try some other virtualization method [37].

4.3.2 Software Virtualization

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

Binary translation is the emulation of one instruction set by another through translation of code. Instructions that are executed by the OS are translated from the source to the target instruction set [34]. This form of virtualization does not require any changes to the virtual operating system. It is a form of virtualization where the VMM reforms the VM's commands for the underlying hardware. The translation of the commands naturally introduces some overhead but can also optimize and gain performance boost with some instructions [4].

4.3.3 Hardware Support

Due to the rise of interest in the virtualization, the processor manufacturers have included virtualization support in their products. Intel VT Vanderpool/Virtual Technology 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 [35]. AMD's Paficifica chip includes AMD-V virtualization support [42].

Both Intel and AMD versions of hardware support try to solve the x86 architecture's problems, described in section 4.3, by adding a separate mode for the guest OS. In this mode the guest OS is able to run on ring level 0 and execute privileged commands as normal [35, 42].

4.4 Xen

Xen started as an open source project and was developed at the University of Cambridge. Then it was producticed by Xensource [3]. Now there is both commercial and open source versions [4] of the project. It has been included into various Linux distributions such as Ubuntu, Suse and Red Hat. Xen VMM has also been included to 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. Xen hypervisors architectural structure is illustrated in Figure 4.4. 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 privileged guest domain called domain0. Domain0 is brought up at boot time and it is able see all the hardware. It contains the actual hardware device drivers and has the tools to manage guest operating systems, the virtual machines. Guest operating systems are given an abstraction of the device drivers, the front end pieces. This lightens the hypervisor and also gives protection against faulty drivers as they are separated from the virtual machine monitor [5].

Though Xen requires modifications to the guest operating system, no changes

[3] `www.xensource.com`
[4] `www.xen.org`

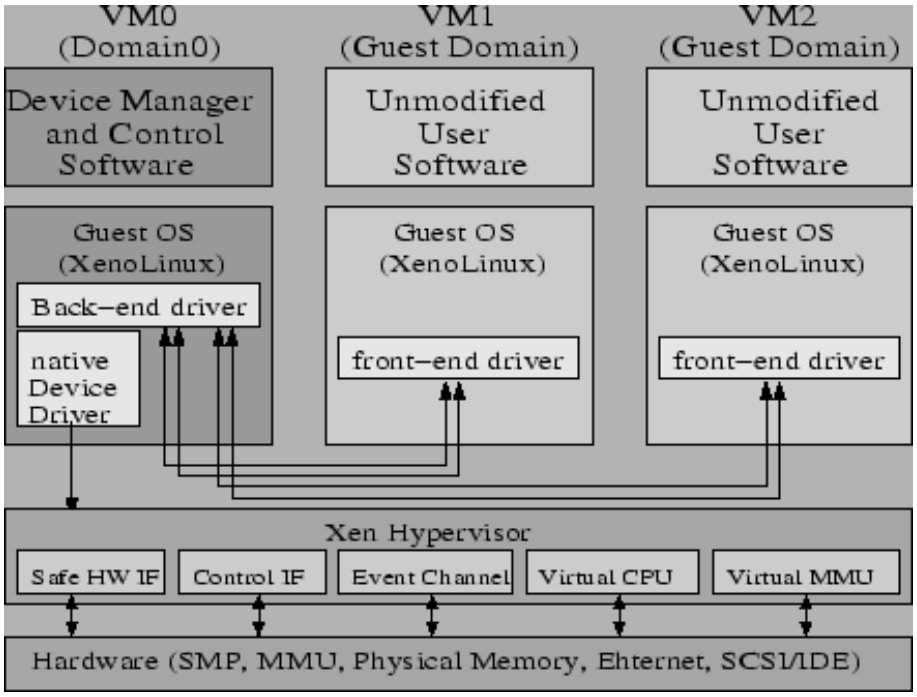

Figure 4.3: Xen hypervisor architecture (Courtesy of [22])

are needed to the application binary interface (ABI). Meaning that the guest applications can run unmodified.

4.5 VMware

VMWare is one of the biggest virtualization solution providers. It offers a wide range of virtualization products. Products such as Workstation for host based virtualization and ESX Server for server based virtualization. Most of their products are proprietary but they also offer some open source products. VMWare uses binary translation in its products but has also support for paravirtualization [36].

4.6 KVM - Kernel Virtual Machine

Kernel Virtual Machine is a newcomer in the virtualization domain. It exploits the recent hardware virtualization enhancements in processors, which were de-

scribed in Subsection 4.3.3. It has been included in the Linux kernel [5] since version 2.6.20.

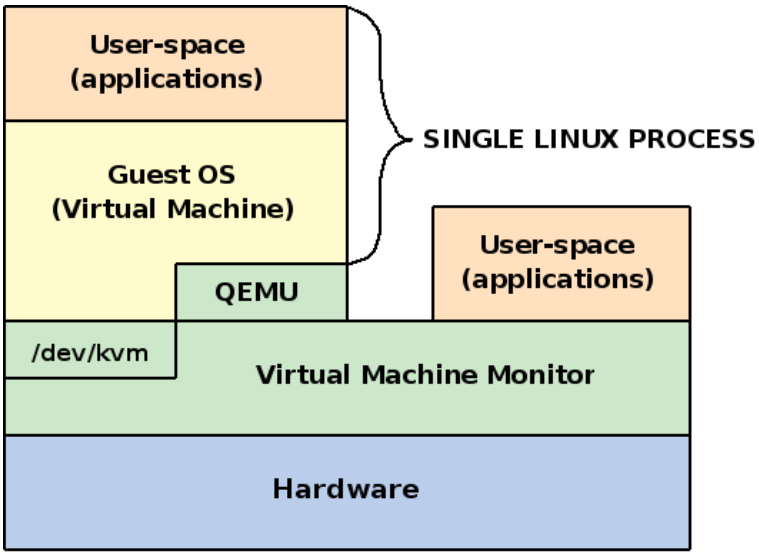

Figure 4.4: Kernel Virtual Machine architecture (Courtesy of [18])

Figure 4.6 illustrates the architecture of KVM. Linux kernel is turned into a virtual machine monitor by adding kvm-module into it. Virtual machines are processes on top of host operating system. This means that KVM may use the scheduling and memory management properties of the vastly developed Linux kernel.

KVM module enables the near to machine speed virtualization of processor and the virtualization of memory inside kernel. I/O devices of the guest operating system is handled by a user space QEMU process. QEMU emulates the I/O for the guest operating systems and provides virtualization of I/O devices.

The two main disadvantages of this otherwise simple and powerfull virtualization technique are the emulation of the I/O devices in user space and the need for special hardware [18]. Especially the emulation of I/O slows down the otherwise fast hardware virtualization.

[5] `kernel.org`

4.7 Linux VServer

Linux VServer [6] is another virtualization technique that takes advantage of the properties build into the Linux kernel. Virtual machines are run in user space as normal programs and they share the operating Linux kernel with the underlying operating system and other virtual servers. Meaning that all the processes of the virtual machines are scheduled by the same kernel.

Vserver virtual machines are contained in their own root and security context. This means that the virtual machines are unaware of the other virtual servers or the underlying operating system. Though VM file system is part of the underlying operating system's file system, they have no way accessing or seeing anything beyond their root. Separate security context means that the virtual machines only see their own processes [12].

4.8 Summary

There are plenty of virtualization solutions even for platform virtualization. They all have their applications and the suitability depends much on the requirements. It can vary from setting up testing environments on a PC to 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 offer good support. Open source products tend to require more knowledge to set up but they are free. User space virtualization solutions such as KVM and VServer offer a powerful virtualization solution that is 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 mahcines with complete operating systems. These VMs have normal access to I/O-devices with unnoticeable overhead but every new virtual machine introduces duplication.

[6] http://linux-vserver.org/

Chapter 5

Service Provisioning

In this chapter we will take a look at a few systems for remote software and system management. The projects covered in this chapter have different approaches and offer different levels of completeness. Some offer a lot of features that are not in the scope of this thesis.

One thing to take into consideration is the mode of communication. Either it is client initiated meaning that it pulls data from the server or it is server initiated meaning that the data is pushed to the client. This has an effect on the complexity of the firewall.

5.1 Debian Package Management System

Debian Package Manager is a powerful software management system and the foundation of many Debian [1] based operating systems. It is used to install, remove, and upgrade software. Software, either in binary or source form, is stored and distributed inside compressed packages. Package manager can use both external or local package sources for installation.

The package management system can be divided into three main building blocks. The first is the package library also called a repository, which contains all 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

[1] `www.debian.org`

package management tools that 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 you have distributions 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 you have 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 in their directories.

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. Operating systems such as Debian or Ubuntu have repositories, which contain thousands of packages. Packages are categorised into distributions 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 their source code. Metadata consists of information such as a version number, package name, author name, packages dependencies etc. Package dependencies determine what other software is required for software to work properly.

The Debian package management system has plenty of tools for software installation, removal and upgrading. All the tools are build on top of the dpkg (Debian 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 front-end the aptitude have the ability to handle these better. APT can be configured to use multiple repositories with source.list. One must be carefull when configuring the sources.list because of the dependencies.

Package management operations are client initiated. First the client fetches all Packages.gz files from the repositories. 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 [7].

5.2 Planetlab

Planetlab is an overlay network of computers. In the early 2008 the Planet lab network consisted of more than 800 computing nodes in over 400 geometrically separated sites [2] so that every continent is covered. This range of distribution provides an exellent testbed for network or distributed computing research.

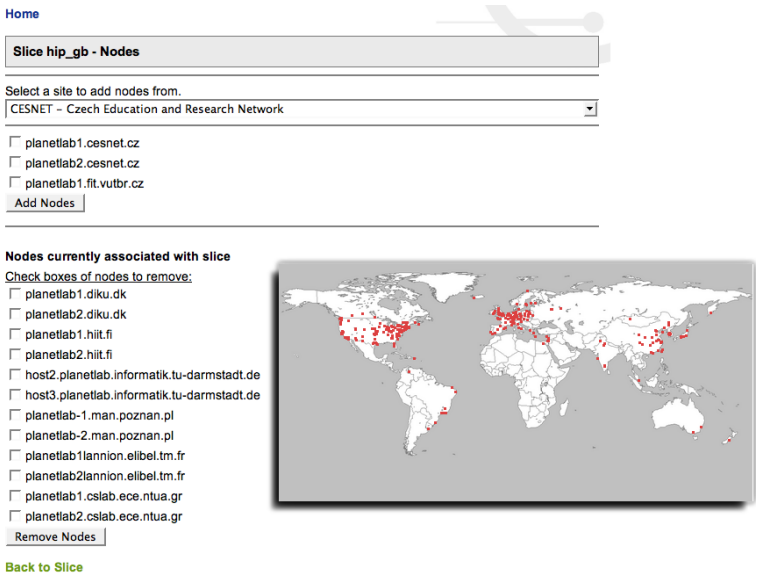

Figure 5.1: Node management interface with the overall picture of node distribution

Figure 5.1 illustrates the magnitude of Planetlab network and the node management interface. Planetlab projects are called slices and they are a set of virtual resources. Slices can have an arbitrary amount of nodes from anywhere in the world and one node can contain several slices. A slice on a node means that the owner of the slice has a active virtual machine running on that node. The virtualization on the node machines are made possible with Vserver virtualization

[2] `www.planet-lab.org`

technique, which is explained in more detail in section 4.7.

The management of slices and nodes is centralized. They are created via The Planetlab central server. Meaning that the final control is in the hands of Planetlab administration[9]. 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 needs to be installed by the client. The Planetlab community provides some tools for the software management of the slices [3].

5.2.1 My Planetlab Central - MyPLC

My Planetlab Central (MyPLC) is the actual software used in Planetlab. With MyPLC one can set up its own overlay networks. MyPLC code is freely usable allowing people to make their own versions of the Planetlab software [1].

5.3 Smart Domains

Smart Domains is a virtual machine resource management infrastructure. It was designed to set up distributed environments for running batch GRID jobs and conducting system tests. Smart Domains builds on Smartfrog framework and Xen virtualization. It provides a way to configure, deploy, monitor, and manage large distributed environments [40].

Smartfrog is a java based software made by HP labs. Smartfrog provides:

- A rich description language for describing resources and their interoperaribility.
- A deployment engine for delivering configurations and software to the p2p network of smartfrog nodes.

Smartfrog uses java Remote Machines Invocation (RMI) for communication between peer to peer nodes. This makes it less applicaple in large scale networks

[3] https://www.planet-lab.org/tools

as firewalls usually block RMI traffic [13].

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

Chapter 6

Solution

The goal of the project was to make a system to set up and replicate SBC services at geographically distributed locations. This is achieved with virtualization. Virtualized services can be easily moved to different locations and started on top of the abstraction provided by virtualization.

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 security layer between different services, which means that compromising one service leaves others intact.

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 6.1.

We use Ubuntu as the base operating system. All software can be found from Ubuntu [1] Edgy and subsequent distributions, which means that all used components are open source. Since we have only used existing software, there will be little or no code to upkeep.

[1] www.ubuntu.com

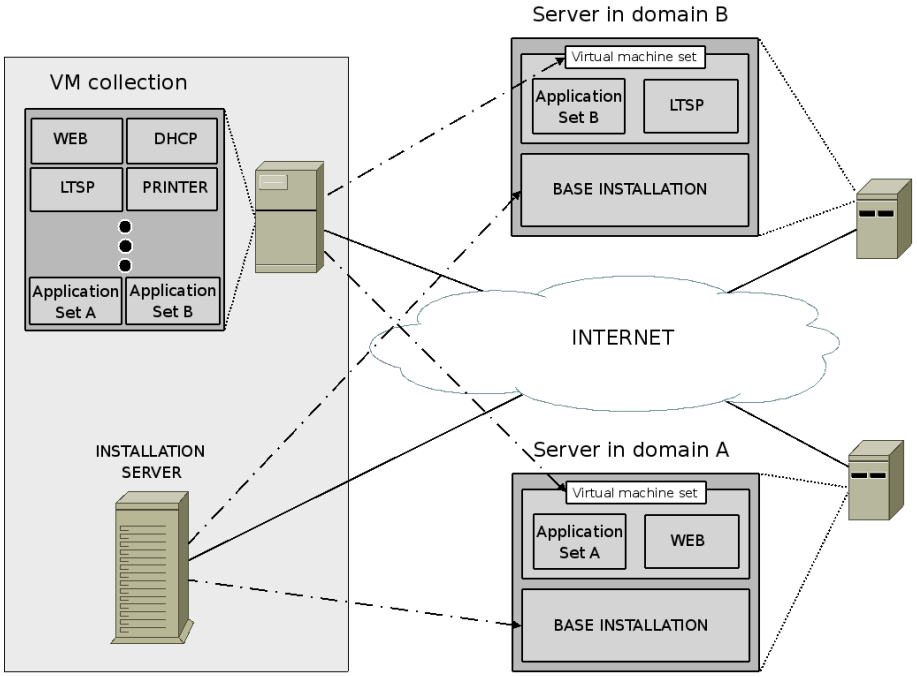

Figure 6.1: Prototype system overview

As our virtualization platform we have chosen Xen and paravirtualization. Xen is open source and it has advanced quite a lot in the past few years. Xen's paravirtualization 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.

6.1 SBC Service

In our system the SBC services are installed into virtual machines. In this simplified SBC system we have separated access and application services to their own virtual machines. The access server is responsible for setting up the thin clients. It is installed with the LTSP environment described in Section 3.2. The application server has all the applications and the desktop environment which thin clients connects to when set up. It is installed with Gnome [2] and some common office tools.

[2] www.gnome.org

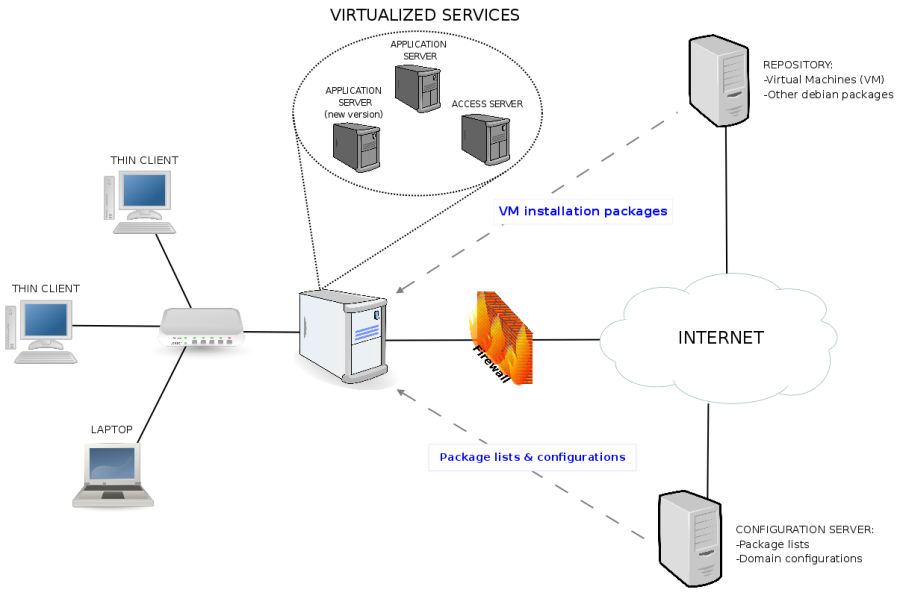

Figure 6.2: Virtualized SBC environment

Figure 6.2 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. This makes the shift from version to another as convenient as possible. In the prototype the physical machine also works as a firewall. Virtual machines are constrained into their own LAN which has the physical machine as the gateway. Depending on the hardware the physical machine the virtual LAN may be connected to a physical LAN, where the thin clients reside, via other network card.

6.2 Operation of system

Work flow in short as illustrated in Figure 6.3:

1. Client machine installation is started using general installation medium.
2. Installation parameters are retrieved from the configuration server.
3. Installation packages are downloaded from a repository and installed.
4. The packagemanager is started and the package list is retrieved from the configuration server.

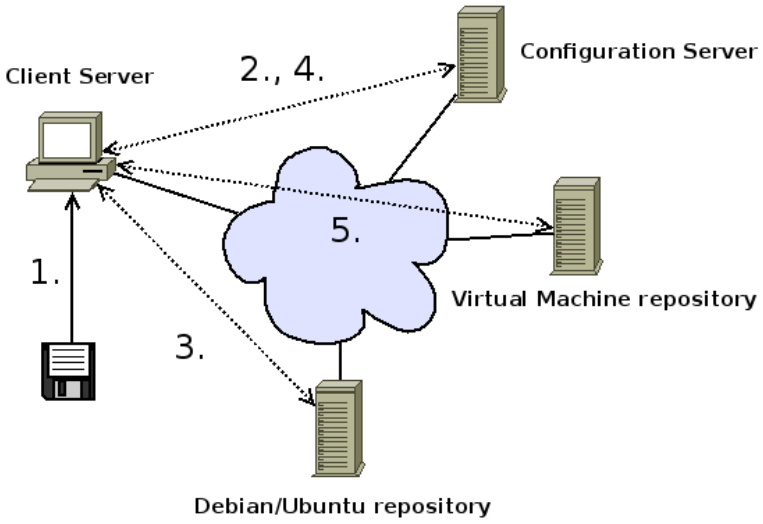

Figure 6.3: Workflow of the system

5. Virtual machine installation packages retrieved from a repository and installed according to the list.

6. Go back to step 4.

The client machines are installed using a remote installation service. The client is given a USB-stick, CD-Rom or a corresponding media, which is used for the installation of its 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 almost automatically. 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 of clients' computers is managed.

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 the base operating system's repository 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.

6.3 System Components

Next we explain the components that comprise the automated installation and management system. The complete installation documentation has been included in the appendice 8.

6.3.1 Client Installation Media

The client installation media is used to install remote machines with the base operating system. Booting a machine with the installation media loads the Linux operating system that handles the installation of the client server. The installation will set up Ubuntu Feisty OS with 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. 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 our experiment in Finland, can be found from the appendix 8. Preseed files are stored and managed on the central server. Centrally located preseed files are easier to update and makes the actual installation media more generic.

6.3.2 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 predifined tree directory structure. The metadata is used by Debian and

Ubuntu installation tools, the apt-tools, to install and manage packages and their dependencies [16].

The repository itself is created and managed with an open source tool called Reprepro [1]. Reprepro manages the versioning of packages and it also signs every package when they are 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.

6.3.3 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 very movable in the network.

Every virtual machine has its own Debian package. Usage of Debian packages makes it easy to manage the files, that are required by the VM, and to script functionality to 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 environment for VM and eventually to start the VM. 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 [1]. It is also the base of Debian package management system. Dpkg is given as parameter a directory that contains all files needed by the virtual machine in a predefined tree structure and in addition a special control file directory. When installing the package the files in the package will be placed according to that structure. The special control file directory contains the metadata of the package and also all the installation and removal scripts.

[1] http://mirrorer.alioth.debian.org/
[1] http://en.wikipedia.org/wiki/Dpkg

6.3.4 Configuration Server

The configuration server is used to store the installation package list of the client. Lists are distributed in pull fashion using the Rsync programme [3]. 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 [17]. Now the traffic is generated from inside and no extra holes are made for the incoming traffic.

6.3.5 Packagemanager

PackageManager is an installation package management tool. It is installed as part of the base installation. Packagemanager retrieves package lists from the configuration server and updates the machines software accordingly. It uses rsync and ssh to communicate with the server. List are retrieved on daily basis and also at boot time.

After retrieving the lists the packagemanager calls pkgsync, which is a package management tool of the operating system. Pkgsync synchronizes packages according to the list. Every package mentioned in the list is installed and missing packages are removed. Also if there are updates to the packages currently installed, they are applied.

6.4 Cern Library Pilot

Cern library needed an easily maintainable and silent system that fits to their needs. The system described in this chapter was used to deliver the library a custom-built Server Based Computing system. Updates to the systems are automatic and the remotely managed. Figure 6.4 illustrates the architecture of the Cern library pilot.

[3] http://samba.anu.edu.au/rsync/

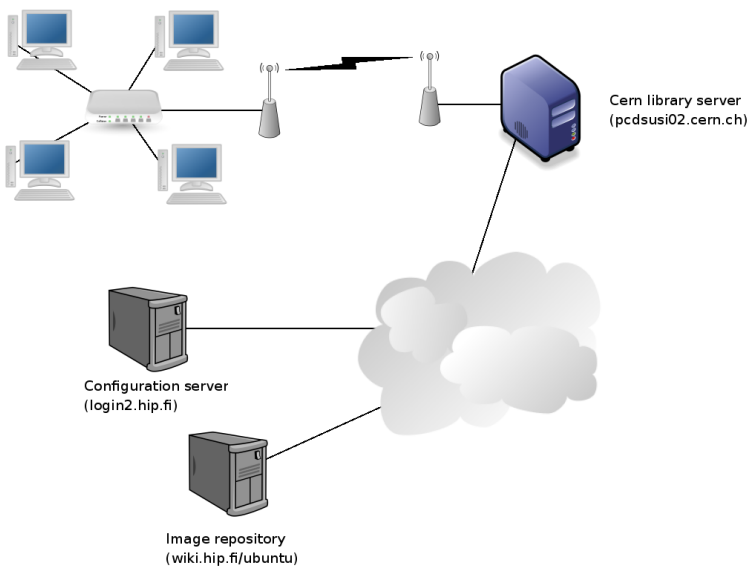

Figure 6.4: Cern library pilot

The library had old dektop PCs from which the server was chosen. The server was installed using our systems. The library also obtained some economical, diskless and power wise limited computers [4], which would nowadays be considered useless for anything else. But as thin clients they work well enough.

The server hosts two virtual machines. One for 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 LTSP environment. The application server is also installed with Ubuntu Feisty on top of which we have installed the xfce4 [5] 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 X window 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. This relieves of the burden of installing cables to and making the stations more movable. Having wireless connection between the server and client also introduces security problems which are now dealt with ssh

[4] `http://www.icoptech.com/products_detail.asp?ProductID=271`
[5] http://www.xfce.org/

encryption.

Chapter 7

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.

7.1 Key Features of Prototype

The solution proposed in this thesis attempts to solve 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 generic network installation system and with the virtualization of services.

In our solution the services are tested and packaged centrally by professionals. Professionals that are able to make the installations more reliable for the end user. This makes the end result more robust and secure as it would be by a person that is starting 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 interface to be added on.

7.2 Usability

As all administrative tasks, such as creation of virtual machines and configuration of client package collection, are handled centrally. The end user has almost no responsibilities. Meaning that the end users could rely on the central administration. This arrangement fits quite well for the category of to people whom the system was designed. Although the setting does not restrict more experienced users to their modifications.

Though clients have rights to modify their system, the changes are not applied to the installation images in the central server, which in part makes the changes temporary. This can also be seen as a good feature as all the damages that the client does can be unmade.

Installation is automated using old and matured Debian Installer. Answers to the questions asked by the installer are preanswered by preseed configuration file, which removes the need for local involvement in the installation process. After the user has put the installation media into the machine and booted the machine the installation system should be able to do the rest.

7.3 Performance

The system relies heavily on high speed network connections. The installation media and configurations are located on remote servers. Though it is possible to have local repositories for images as well, it would make their management more complex. Having a centralized server makes the management more scalable as the same configurations and images can be reused in multiple locations. Also, the use of virtualization allows us to use the existing resources more thoroughly as normally servers run services with very light load.

7.3.1 Distribution of Services

In our solution the virtual machines are distributed inside image files that are several GBs in size. These files contain complete Linux root file systems (FS). The actual size of the image of FS does not so much matter as the empty space compresses more than thousand-fold. This makes the FS size indifferent and makes the transfer of the virtual machines with large file systems feasible.

There are several ways to provide the virtual machine FS [8]. One can use physical partitions, lvm logical volume manager, network file systems, or 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 client side. The relevance of the disk performance is not that big and loopback devices perform relatively well. In our tests we have noticed that the loopback disk images perform at application level as well as, or even better, than the actual physical disks.

Figures 7.1 and 7.2 illustrate the transfer speeds gathered from our test environment where we had two otherwise identical virtual machine running except the other had its fs on physical partition and the other on a loopback device image. Figure 7.1 illustrates the transfer speeds of cp and dd commands inside the virtual machines and inside one fs. Figure 7.2 illustrates the read and write transfer speeds between virtual machine and another server in a high speed local area network. Small files 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.

Table 7.1: Local transfer speeds in VMs

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 7.2: SCP transfer speeds to VMs in high speed LAN in MB/s

SCP to asynchronous disks				
	$Physicaldiskpartition$		$Image$	
	$Bigfilewrite$	$Smallfileswrite$	$Bigfilewrite$	$Smallfileswrite$
No load	5.91	0.21	11.91	0.21
Load	0.63	0.09	7.30	0.17
	$Bigfileread$	$Smallfilesread$	$Bigfileread$	$Smallfilesread$
No load	28.09	0.22	27.47	0.22
Load	4.63	0.10	4.45	0.08
SCP to synchronous disks				
	$Physicaldiskpartition$		$Image$	
	$Bigfilewrite$	$Smallfileswrite$	$Bigfilewrite$	$Smallfileswrite$
No load	5.64	0.20	13.06	0.20
Load	1.14	0.17	8.67	0.19
	$Bigfileread$	$Smallfilesread$	$Bigfileread$	$Smallfilesread$

No load	23.51	0.22	26.27	0.22
Load	6.86	0.15	5.49	0.13

7.3.2 Security

The most important advantage with images 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.

Virtual machines are distributed using repositories. It would be possible to eavesdrop the traffic between 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 centralized maintenance of images makes their management more scalable as one image can serve several clients. It also makes the services more secure as this way the work will be done by a professionals with accumulated experience on the subject. The network of clients gather data that the centralized administration can then use to enhance the services. This is a security issue as it is important that the programs are well configured and up to date.

The public key of the repository is fetched by the automated installation via HTTP, meaning that it is insecure. It might be possible to use HTTPS for this but it would require the use of certificates to make the server trustworthy for the client. Certificates tend to expire so they make the installation media less generic.

7.3.3 Virtualization

Virtualization makes it possible to multiplex the resources of one machine among several virtual machines. This makes it possible to set up several services into one machine without compromising the others. Having services in their own machines create a new layer of security as there is no actual link between machines except probably a network connection.

Among several different virtualization solutions we chose to use the Xen project. Xen uses paravirtualization that offers high performance in an open source package. With paravirtualization we get fast I/O performance which is not reduced by emulation from which most other solutions suffer from. Virtual machines do not run in the kernel space so the virtualization platform is quite well protected. Xensource has also shown active development in the past years and offers a multitude of tools for management of virtual machines and pools of servers. VMWare would have corresponding virtualization products but its licencing and proprietary nature makes it less appealing for our use

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

7.4 Serving SBC

So far the discussion has been mostly about generic services. Now we will take a look at how the system serves Server Based Computing. SBC, as described in the chapter 3, is a system where all the applications, processing and storage space is centralized. It can be used to offer users applications, operating systems and devices. All of which can be accessed via thin client middleware.

In our prototype we have separated access service from the application service. Also the authentication and firewalling is separated. All the services are installed 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. The other offers a booting media for the thin clients on the LAN and the other desktop environment and applications. Now that 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 and services within.

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 are able to 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 raising up of new services in the fast lan even faster.

7.5 CERN Library Use-case

The CERN library was used a test environment for our solution. Idea was that the system would relieve the librarians from their administrative tasks and provide a usable and secure system for the library. We will compare in this section the pilot system with the existing alternative, the windows terminals.

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

7.5.1 Performance

While the installation of one Windows terminal from the CERN windows domain takes more than an hour, the installation of one Xen server with virtual machines required by the thin client system takes less than an hour. The set up time of one thin client is about 5 minutes. This means that the set up time of one thin client is less than that of the installation of one Windows terminal. Table 7.3 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 see the time savings achieved.

Table 7.3: Terminal set up times

	SBC	Windows terminal
Time components:		
HW installation / pc	5 min	5 min
Base system	11 min	45 min
Virtual machine	>15 min	-
Office capability	35 min	1 h 35 min
Total time:		
Installing 1st terminal	56 min	2 h 25 min
Setting up a new terminal	5 min	2 h 25 min
Reinstalling services	15-56 min	-

One thing to remember is that both systems download installation media from servers making them both dependant on the network. Both systems require updates from time to time. Some installations 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 reinstalled. With our system it is enough to just update the virtual machine serving the thin clients. This can be done so that almost no downtime occurs.

7.5.2 Terminal Devices

Windows requires much more powerful terminals than those of the thin client system. Usually they also have a lot of wearing parts that need replacing. At some point these parts become unavailable, which forces the renewal of the whole terminal. Thin client can be just about any machine with the right architecture, which x86 in our case. The simplest thin clients have no wearing parts such as disks or fans. Also the thin client size is far more smaller than that of normal work station making them almost unnoticeable.

Machines without any wearing parts have longer life span. Meaning that the need to replace old clients with newer ones come less seldom. Having no fans or disks also mean that there are no noise producing parts in the machine. Full

blown Windows terminals need a heavy duty machine even to perform a simple task. These machines produce a lot of heat, which means they need cooling. The smaller the actual device gets the harder it is to cool quietly.

7.5.3 Open source

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

7.6 Discussion

141-x-143

The system is easy to build and 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 reliesed. 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 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 more 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.

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.

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 8

Conclusions

In this thesis we studied server based computing, virtualization, and techniques of service distribution. This research was done as part of the Netgate 2 research project. We developed a system that automates the installation of remote servers with virtualized services. This system has been tested in several locations such as the CERN library.

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

Virtualization allows us to package services and create a modular and secure system. As most of the services can be found from Linux distributions we can use Xen and paravirtualization for virtualization. Paravirtualization induces less overhead than other virtualization solutions that need to emulate or translate machine instructions. For the distributions of our virtualized services we chose Debian package management system. It is a mature system that has proven functional. It offers all the tools we need 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, teletravail etc. The thing in common to all of them is centralization of resources and simplification of end devices. Making the addition of a new client as simple as possible. The applicability of SBC has been restricted by weak network infrastructure and expensiveness of server hardware. Today networks are fast

and processing power inexpensive. Now the facilities for SBC are there and it is gaining momentum.

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] 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.

[5] 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.

[6] H. K. Bjerke. Hpc virtualization with xen on itanium. Master's thesis, Norwegian University of Science and Technology, July 2005.

[7] D. Blackman. Debian package management, part 1: A user's guide. *Linux J.*, 2000(80es):12, 2000.

[8] P. Chaganti. *Xen Virtualization, A practical handbook*. Pact Publishing Ltd., 2007.

[9] 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.

[10] C. L. Coffing. An x86 protected mode virtual machine monitor for the mit exokernel.
Referenced 21.1.2007.

[11] H. D. Company.
hewlett-paccard 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.

[12] 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.

[13] 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.

[14] S. Greenberg. What is thin client computing. *For the Record*, July 2000.

[15] S. R. K. M. S. Information Society Structures Committee, Chair: Erkki Salmio. Information society structures in educational institutions - results of the surveys 2004 and summary of the years 2004. Technical report, Ministry of Education, 2004.

[16] A. Isotton. Debian repository howto.
`http://www.debian.org/doc/manuals/repository-howto/repository-howto`.

[17] J. P. John Wack, Ken Cutler. Guidelines on firewalls and firewall policy. Technical report, National Institute of Standards and Technology, 2002.

[18] M. T. Jones. Discover the linux kernel virtual machine.
`http://www.ibm.com/developerworks/linux/library/l-linux-kvm/`,
April 2007.

[19] J. Kanter. *Understanding Thin-Client/Server Computing*. Microsoft Press, 1998.

[20] 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.

[21] A. Kros and M. A. Margevicius. Thin-client shipments stage strongest growth since 2000. Technical report, Gartner, June 2006.

[22] 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.

[23] 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.

[24] Microsoft. *Remote Desktop Protocol (RDP) Features and Performance*. `http://www.microsoft.com/technet/prodtechnol/Win2KTS/evaluate/featfunc/rdpfperf.mspx`.

[25] Microsoft. *Configuring authentication and encryption*, January 2005. `http://technet2.microsoft.com/windowsserver/en/library/a92d8eb9-f53d-4e86-ac9b-29fd6146977b1033.mspx?mfr=true`.

[26] S. 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.

[27] I. M.Revett and C.Stephens. Network computing: a tutorial review. *Electronics & Communications engineering journal*, February 2001.

[28] S. R. (Nomachine). Getting started with nx. `http://www.nomachine.com/documents/getting-started.php`. Referenced 7.5.2008, Last edited 1.8.2007.

[29] A. Nye. *X Protocol reference Manual for X11 Version 4, Release 6*. OŔeilly & Associates, inc, 1995.

[30] N. plc. Server based computing explained. `http://www.netvoyager.co.uk/general/sbce.html`. Referenced 21.8.2006, Last edited.

[31] T. Richardson. The rfb protocol. Technical report, RealVNC Ltd, 2007.

[32] R. Rose. Survey of system virtualization techniques. citeseer.ist.psu.edu/rose04survey.html, 2004.

[33] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.

[34] A. Tijms. Binary translation: Classification of emulators. Technical report, Leiden Institute of Advanced Computer Science, 2000.

[35] 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.

[36] VMWare. Technology preview for transparent paravirtualization. `http://www.vmware.com/interfaces/techpreview.html`. Referenced 16.5.2008.

[37] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble. Rethinking the design of virtual machine monitors. *Computer*, 38(5):57–62, 2005.

[38] 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`.

[39] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications, 2002.

[40] S. J. I. D. P. T. Xavier Grehant, Olivier Pernet. Xen management with smartfrog.

[41] 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.

[42] A. Zeichick. Processor-based virtualization, amd64 style, part ii. `http://developer.amd.com/article\_print.jsp?id=15`. Referenced 3.11.2007, Last edited 30.6.2006.

Remote management system installation instructions

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 MBR that can be found from the repository. They are contained insidebootingtools.tar.gz, which can be found from the root of the repository, and inside dom0config package.

Installation tool is configured to boot with predifined 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 becessary installation medias. There you can also define your your own repository that contains your virtual machines. You musta 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 preseed. 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

descibed 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 afterwords 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. Meaning that file A in folder B under building folder will be placed into /B/A 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/¡name 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 prerm 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 reguired 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

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 server the packagelists. The server has a client specific account which only accessible with ssh+rsync. Meaning that 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

Dom0config package configures the xen dom0 which is the priviliged den 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
```

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 in 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 1. Virtual lan in the figure illustrates the network inside pcdsusi02.

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 clus-

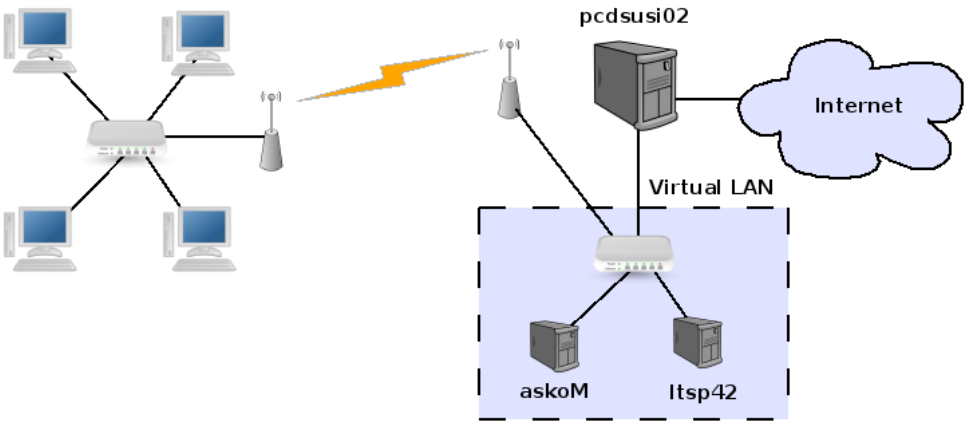

Figure 1: Thin clint system's network

ters 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 tour 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 assemplies: Xen virtualization software, LTSP thin client managent 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 heterogenouos 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.