

Jukka Kommeri

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

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

HELSINKI UNIVERSITY  
OF TECHNOLOGY

ABSTRACT OF THE  
MASTER'S THESIS

<b>Author:</b>	Jukka Kommeri	
<b>Name of the thesis:</b>	System management in Server Based Computing with virtualization	
<b>Date:</b>	24 April, 2007	<b>Number of pages:</b> 32
<b>Department:</b>	Electrical and Communications engineering	
<b>Professorship:</b>	S-38	
<b>Supervisor:</b>	Professor Jörg Ott	
<b>Instructor:</b>	Tapio Niemi, Dr, Helsinki Institute of Physics	
Some text...		
<b>Keywords:</b>	sbc, virtualization, xen, ltsp	

TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

<b>Tekijä:</b>	Jukka Kommeri
<b>Työn nimi:</b>	Palvelun hallinta palvelin keskeisessä ympäristössä
<b>Päivämäärä:</b>	24. huhtikuuta 2007 <b>Sivuja:</b> 32
<b>Osasto:</b>	Sähkö ja tietoliikennetekniikan osasto
<b>Professuuri:</b>	S-38
<b>Työn valvoja:</b>	Professori Jörg Ott
<b>Työn ohjaaja:</b>	Tapio Niemi, Dr, Helsinki Institute of Physics
Tiivistelmä...	
<b>Avainsanat:</b>	sbc, virtualization, xen, ltsp

# Acknowledgements

Some text...

CERN Geneva, 24 April, 2007

Jukka Kommeri

# Table of Contents

<b>Contents</b>	<b>vi</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>Abbreviations</b>	<b>ix</b>
<b>Abbreviations</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Problem</b>	<b>3</b>
2.1 Netgate 2 . . . . .	3
2.2 Research Problem . . . . .	4
2.3 CERN library use-case . . . . .	4
<b>3 Server Based Computing</b>	<b>6</b>
3.1 Existing thin client middleware . . . . .	7
3.1.1 ICA - Independent Computing Architecture . . . . .	7
3.1.2 RDP - Remote Desktop Protocol . . . . .	7
3.1.3 VNC - Virtual Network Computing . . . . .	8
3.1.4 X Window System . . . . .	8
3.1.5 NX - New X . . . . .	8
3.2 Linux Terminal Server Project - LTSP . . . . .	9
<b>4 Virtualization</b>	<b>10</b>
4.1 Overview . . . . .	10
4.2 Why virtualize . . . . .	11
4.3 Methodology . . . . .	12
4.3.1 Paravirtualization . . . . .	13
4.3.2 Software virtualization . . . . .	14
4.3.3 Hardware support . . . . .	14
4.4 Xen . . . . .	14
4.5 VMware . . . . .	15
4.6 KVM - Kernel Virtual Machine . . . . .	16

4.7	Linux VServer . . . . .	17
4.8	Summary . . . . .	17
<b>5</b>	<b>Provisioning</b>	<b>18</b>
5.1	Debian package management system . . . . .	18
5.2	Planetlab . . . . .	20
5.2.1	MyPLC . . . . .	20
5.3	Smart Domains . . . . .	21
<b>6</b>	<b>Solution</b>	<b>22</b>
6.1	Operation of the system . . . . .	23
6.2	System components . . . . .	24
6.2.1	Client installation media . . . . .	24
6.2.2	Image server . . . . .	25
6.2.3	Image . . . . .	25
6.2.4	Configuration server . . . . .	26
6.2.5	Packagemanager . . . . .	26
6.3	Cern library pilot . . . . .	26
<b>7</b>	<b>Results</b>	<b>28</b>
7.1	CERN library use-case . . . . .	28
7.1.1	Installation . . . . .	28
7.1.2	Management . . . . .	28
7.1.3	Comparison . . . . .	28
7.2	Performance . . . . .	28
7.2.1	Image transfers . . . . .	28
7.2.2	Application memory usage . . . . .	28
<b>8</b>	<b>Conclusions</b>	<b>29</b>
	<b>Configuration files:</b>	<b>33</b>
.1	Preseed file . . . . .	33

# List of Figures

3.1	simple X window system architecture . . . . .	9
4.1	Virtualization architecture . . . . .	11
4.2	x86 ring model (Courtesy [2]) . . . . .	13
4.3	Xen hypervisor architecture (Courtesy [21]) . . . . .	15
4.4	Kernel Virtual Machine architecture (Courtesy [18]) . . . . .	16
5.1	Node management interface with the overall picture of node distribution	20
6.1	Prototype system overview . . . . .	23
6.2	System's workflow . . . . .	23

# List of Tables

# Abbreviations

APT	Advanced Packaging Tool
CERN	l'Organisation Européenne pour la Recherche Nucléaire
GDI	Graphics Device Interface
HIP	Helsinki Institute of Physics
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
RFB	Remote Frame Buffer
RDP	Remote Desktop Protocol
SBC	Server Based Computing
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 where all the applications are installed, managed and executed on a server [27]. 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.

SBC itself is nothing new. It has been around for many decades. It has been a good solution for many use but not a complete solution due to its dependency on networks and computing power. Today the computing power is quite cheap and the networks are getting faster. Now it can be considered as an actual alternative for PCs.

SBC systems are easier to administer because everything is on one machine. The hardware requirements for client side computers are very minimal. Basically old computers with network cards are enough. The centralization also increases the overall security because only the administrators are allowed to configure and install new components.

Interest on this topic has not only been academic. There are several cases where big companies have adopted this technology. For example Finnish weather forecasting company Ilmatieteen laitos has just announced to replace half of its desktops with thin clients. The University of Tampere has been using it for many years. According to Gartner's studies [20] the thin client market has increased 38% last year which is according to them the highest since the year 2000. There are some commercial SBC solutions such as the Sun Microsystems' Sun Ray [24], HP's Server Based Computing solution [11] and a new comer the Panologic <sup>1</sup> as well as non commercial ones such as LTSP <sup>2</sup>, K12LTSP <sup>3</sup>.

---

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

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

<sup>3</sup>[www.K12LTSP.org](http://www.K12LTSP.org)

The problem is that one LTSP server can only serve a limited number of users. In a big organization it may require 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 image distributor. A server for servers that could be used to set up the SBC system in geographically dispersed sites. It would have separate images for different services and needs. This server would make it possible to replicate systems so that the same environment could be set up in various locations.

This thesis is structured as follows: The first chapter will look into different SBC technologies. In the following chapters some virtualization and distributed system management technologies will be covered. Following these related work chapters the actual solution and prototype will be explained in the solution chapter. Its properties will be further analyzed in the following results chapter. Finally we we will summarize the thesis with the conclusion chapter.

# Chapter 2

## Problem

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 <sup>1</sup> is a research project funded by Finnish National Technology Agency (TEKES) and Finnish industrial partners. Research 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 will be 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 research will try to find a way to simplify the installation and management of remote systems. Making it possible for a nontechnical person to set up complicated services on their premises and also relieving him from the administration. This will decrease overall administration costs and increase the quality

---

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

of administration for the end systems i.e. the new systems will be more robust and secure.

## 2.2 Research Problem

The problem 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 really qualified to do the installation but is selected to the task because he has the best knowledge about the subject [15]. This can lead to unstable and unsecure systems. To be sure that the systems work the way that they should, they need to be installed and administered by professionals. Problem is that the usage of professionals is expensive and so is having your own administration.

To make the services available in working environments that lack the required skills, it would help to have a out of the box solution. A solution that outsources the administration. A system that makes it possible to distribute required services to different locations. Services such as sbc, firewalls, webservers, fileserver etc. These services could be configured by professionals and tested before introduction. One would only need to manage one image of every service. These images could then be used to set up systems at different locations. This way the amount of administration would be reduced to the management of images on one server. Client could pick the set of services that is needed and then just start his/her server. The server would download all the required images from the image server and then launch the services inside them automatically.

We plan to solve this by using virtualization and automated installation of the client server. Services could be contained inside virtual machine images and these virtual machine images could then be delivered to the client machines. Virtualization would make it be possible to run different versions of the same system in parallel to test whether the new version works the way it should. This kind of feature would certainly be important in the case critical services.

## 2.3 CERN library use-case

CERN is the world's largest particle physics laboratory <sup>1</sup>. It is situated on the border of France and Switzerland just next to Geneva. It employs nearly 3000 permanent employees and hosts some 6500 visiting scientists yearly. The mission of the CERN

---

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

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 they are using desktop computers that have Windows XP installed and connect to CERN windows domain. Installation time for one of these machines can take about half 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 interferes with the normal responsibilities.

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 and 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 is also done automatically. All the system components are open source, which offers more 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 adding them to the same network and 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 has a positive effect on to the atmosphere of the library.

## Chapter 3

# Server Based Computing

Local area networks and wide area networks have made it possible to centralize services. Instead of having all the 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 the 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 that now the servers are a lot smaller and offer users more usable interfaces [25].

Idea behind Server Based Computing is to move the computation from client's PCs to centralized computing resources. Only thing the PC does is maintain connection to the server, show graphics and transmit information from its peripherals such as keyboard, mouse or USB-drive. This kind of PC is called a thin client[27].

Thin client is no 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. The lifecycle 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 the client does not need to support the higher level graphics which makes it easier to implement. Some protocols adjust to the network by sending updates less frequently and using caching and compression[37].

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 difficult. Below is an introduction to a few popular protocols that are used in different scenarios.

### 3.1.1 ICA - Independent Computing Architecture

ICA 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 lightweigh protocol that is usable in low bandwidth networks such modem connections. It can be used to share both Windows and Unix based operating systems. [19]

### 3.1.2 RDP - Remote Desktop Protocol

RDP 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 [4].

RDP sessions can be configured to meet various needs. It supports different levels of encryption. Traffic between client and server can be secured with transport layer security (TLS) [3] 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 support roaming disconnect which means that the session stays alive though the connection dies [23].

RDP clients are made for Windows platforms. Though there is open source clients for X window systems such as rdesktop <sup>1</sup> and RDP-servers such as xrdp <sup>2</sup>.

### 3.1.3 VNC - Virtual Network Computing

VNC is a open source thin client protocol. It is based on Remote Frame Buffer (RFB). RFB is a thin client protocol that works on framebuffer 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. [28].

VNC represents the 2D graphics and raw graphics end of the protocols. Client retrieves pixel information from the server. Because of this its clients are easier to implement and does not need a heavy system such as the X Window System to work. Due to this fact it can be used in many lighth devices such as mobile phones and personal digital assistants. [37]

### 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 a server that handles input from keyboard and mouse and output to monitor. Clients can be on the same machine or on remote machine. Client and server use X protocol to communicate with each other. [26].

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[37].

### 3.1.5 NX - New X

Tuolta löytyy hyvä kuva: <http://www.nomachine.com/documents/getting-started.php>

---

<sup>1</sup>[www.rdesktop.org](http://www.rdesktop.org)

<sup>2</sup>[xrdp.sourceforge.net](http://xrdp.sourceforge.net)

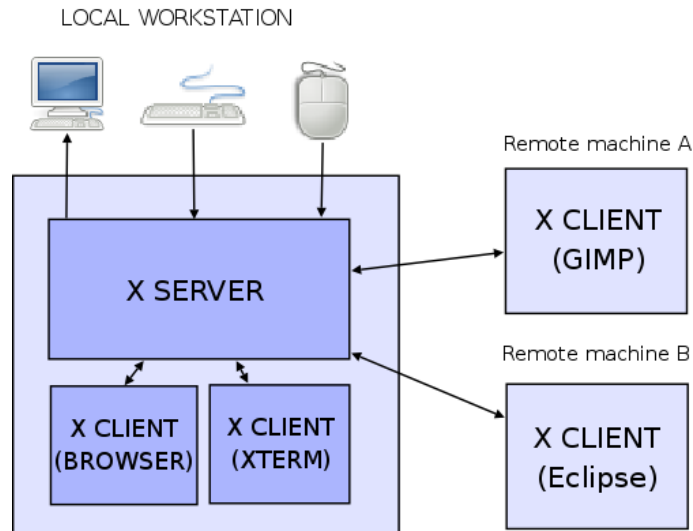


Figure 3.1: simple X window system architecture

### 3.2 Linux Terminal Server Project - LTSP

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<sup>3</sup>. At the time of writing of this thesis the LTSP is in version is 5.0 codename MueKow<sup>4</sup>.

LTSP is an open source project that combines many other open source projects into a comprehensive sbc solution. It offers various ways to connect thin and fat clients to a linux server from different networks. Main usage of ltsp is to connect LAN's workstations 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 the 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. [22].

<sup>3</sup>[www.k12ltsp.org](http://www.k12ltsp.org)

<sup>4</sup>[wiki.ltsp.org/twiki/bin/view/Ltsp/MueKow](http://wiki.ltsp.org/twiki/bin/view/Ltsp/MueKow)

## 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 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 dependent 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 got introduced in the era of mainframes. Then it 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[30].

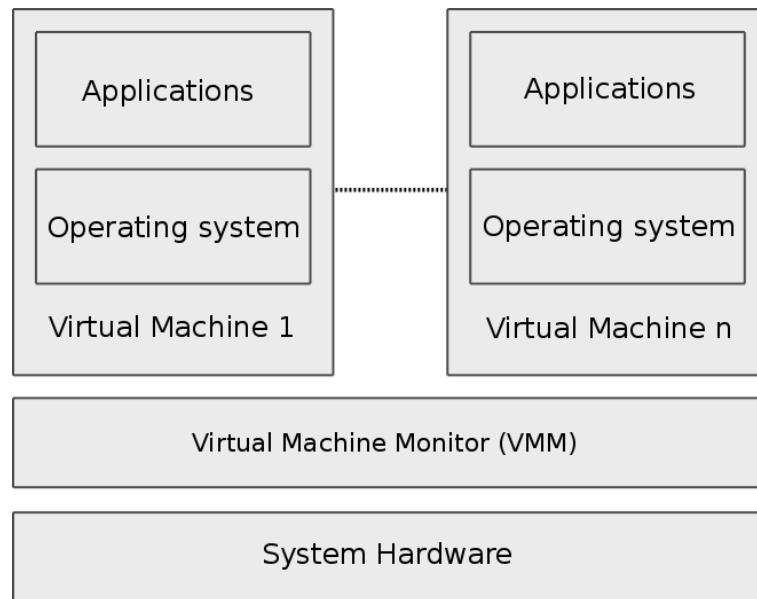


Figure 4.1: Virtualization architecture

With virtualization you can divide the physical hardware among several 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 when installed on their own machines [29].

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 is easy to set up for example testing environments.

## 4.2 Why virtualize

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 downtimes. To have better fault tolerancy, the applications should be distributed among several either physical or virtual machines. The need

for fault tolerancy and the inexpeciveness 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[30].

Virtualization allows us to enclose applications to their own environments. Having applications on separate machines increases the overal 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 on to gain administrative priviliges 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 [35].

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 processor's instructions. The right to execute certain instructions depends on from which level it was executed [10]. Figure 4.3 illustrates the ring structure, 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 [32].

There are many solutions to x86 architectures deficiencies. Here we will introduce three main categories which are used in most of the current virtualization solutions.

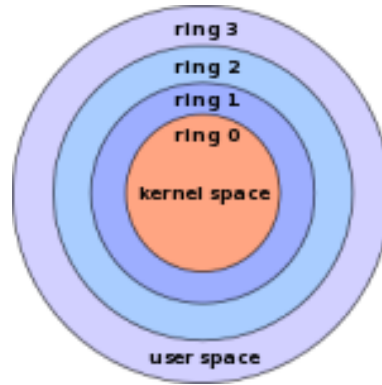


Figure 4.2: x86 ring model (Courtesy [2])

### 4.3.1 Paravirtualization

Architecture used with paravirtualization was originally developed by IBM and utilized in the VM operating system <sup>1</sup>. The term paravirtualization was introduced at 2001 [34] by Denali group <sup>2</sup> that uses paravirtualization in its OS. 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 is the same. Paravirtualization is a technique which 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 to the guest operating system 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 [35].

Problem with paravirtualization is that the owners of the proprietary operating systems might not be willing to modify their OS. Some OS instructions need to be modified so that the virtual machine can operate on a higher privilege level [7]. 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 [33].

---

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

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

### 4.3.2 Software virtualization

Software virtualization is set of virtualization techniques that are used to provide full virtualization. Names of the techniques vary depending on the level they are all 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[31]. 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[5].

### 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[32]. AMD's Pacifica chip includes AMD-V virtualization support[38].

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 guest OS. In this mode the guest OS is able to run on ring level 0 and execute privileged commands as normal. [32, 38]

## 4.4 Xen

Xen started as an open source project and was developed at the University of Cambridge. Then it was producticed by Xensource<sup>3</sup>. Now there is both commercial and open source version<sup>4</sup> 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.

---

<sup>3</sup>[www.xensource.com](http://www.xensource.com)

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

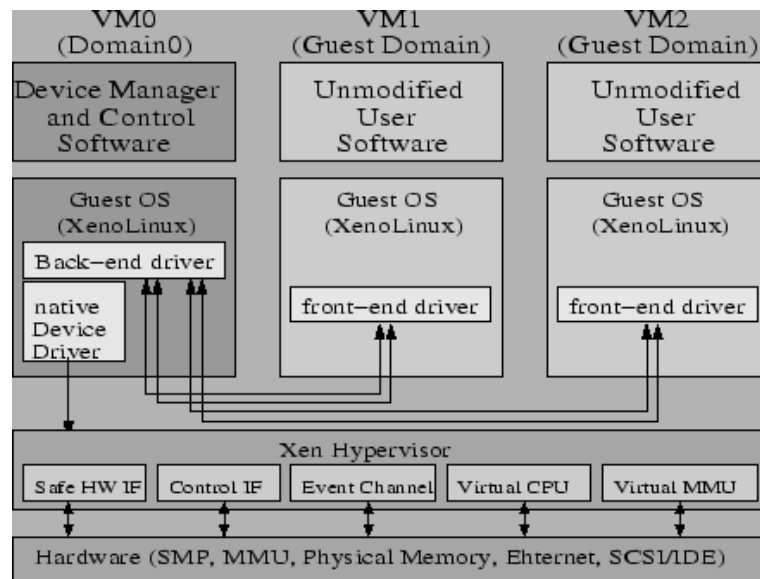


Figure 4.3: Xen hypervisor architecture (Courtesy [21])

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 to see all the hardware. It controls 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 driver, a frontend piece. This lightens the hypervisor and also gives protection against faulty drivers as they are separated from the virtual machine monitor. [6]

Though Xen requires modifications to the guest operating system, no changes 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. These open source products come with some restrictions. VMWare uses both binary translation and para-virtualization in its products.

## 4.6 KVM - Kernel Virtual Machine

Kernel Virtual Machine is a newcomer in the virtualization domain. It exploits the recent hardware virtualization enhancements in the processors, which were described in the subsection 4.3.3. It has been included in the Linux kernel <sup>5</sup> since version 2.6.20.

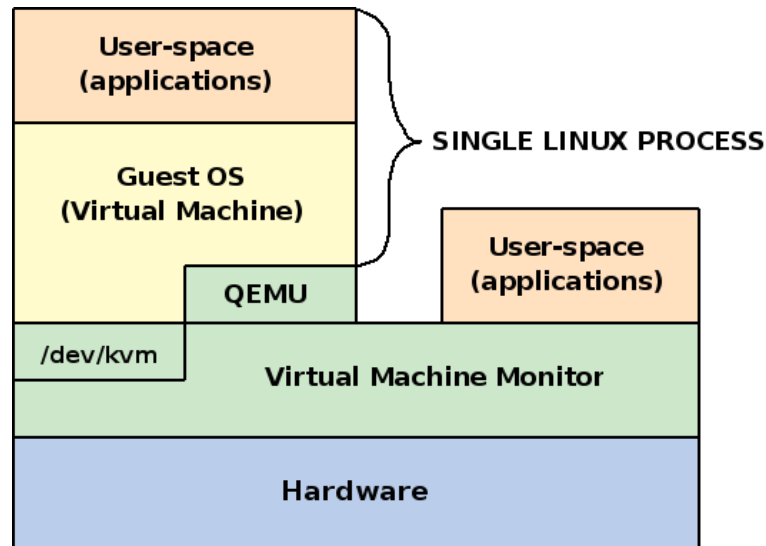


Figure 4.4: Kernel Virtual Machine architecture (Courtesy [18])

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 powerful 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.

---

<sup>5</sup>kernel.org

## 4.7 Linux VServer

Linux VServer <sup>6</sup> 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 filesystem is part of the underlying operating system's filesystem, 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 expensive-ness, easiness of adaptation, efficiency, technical limitations and security. Proprietary virtualization solutions are more expensive than open source versions but they 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 VServers 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 machines with complete operating systems. These VMs have normal access to I/O-devices with unnoticeable overhead but every new virtual machine introduces duplication.

---

<sup>6</sup><http://linux-vserver.org/>

# Chapter 5

## 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 <sup>1</sup> 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.

Package management system can be divided into three main building blocks. One is the package library also called a repository, which contains all the software packages with different versions of them. Second component is the package in which the software is enclosed. Third part contains the client side package management tools that are used to install, remove and update software on the client.

Repository is actually just a collection of files in a predefined directory structure. Directory structure can be of your own design but must be remembered when configuring client

---

<sup>1</sup>[www.debian.org](http://www.debian.org)

side package sources. 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 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 its packages can be signed with a private key. These signed repositories and packages can then be verified with the public key by the user.

Package is the container for the 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 version number, package name, author name, packages dependencies etc. Package dependencies determine what other software is required for software to work properly and can be version specific.

Debian package management system has plenty of tools for software installation, removal and upgrading. All the tools are build on top of dpkg (Debian GNU/Linux Package Manager) tool. Dpkg is a powerfull package management tool but lacks the ability to handle dependencies. APT (advanced packeting 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 [8].

Operation of the system is client initiated. First the client fetches all the 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.

## 5.2 Planetlab

Planetlab is an overlay network of computers. Consisting of more than 800 computing nodes in over 400 geometrically separated sites <sup>2</sup> so that every continent is covered. This range of distribution provides an excellent testbed for network or distributed computing research.

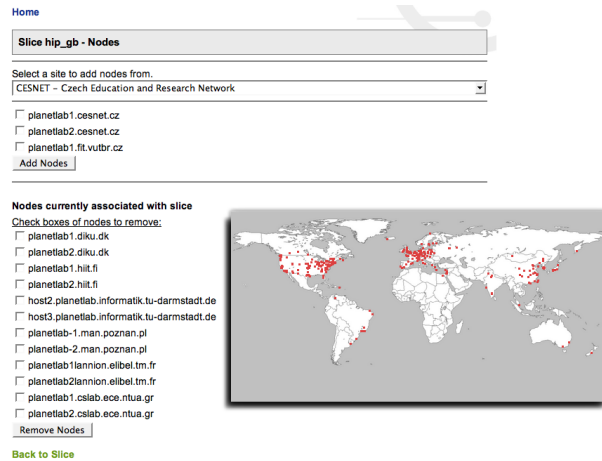


Figure 5.1: Node management interface with the overall picture of node distribution

Figure 5.2 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. Slice on a node means that the owner of the slice has an active virtual machine running on that node. The virtualization on the node machines are made with vserver virtualization technique, which is explained in more detail in section 4.7.

The management of slices and nodes is centralized. They are created via 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 server.

### 5.2.1 MyPLC

MyPLC (My Planetlab Central) is the actual software used in Planetlab. With MyPLC one can set up its own overlay network. [1]

<sup>2</sup>[www.planet-lab.org](http://www.planet-lab.org)

### 5.3 Smart Domains

-xen virtuaalikoneiden hallinta hajautetussa ympäristössä

-smartfrog, Smart Framework for Object Groups -framework konfiguraatioiden ja ohjelmien siirtäminen ,ohjelma voi hakea tarvitsemansa tiedostot tiedostopalvelimelta, ja käynnistäminen reaaliaikaisesti. -ohjelmien välinen kommunikaatio ja riippuvuudet -rikas kieli -RMI kommunikaatioon -With SmartFrog you can -describe deployments -instantiate them across a network -host components that form the application

[36] [13]

## Chapter 6

# Solution

The goal of the project was to make a system to set up and replicate SBC services at geographically distributed locations. We plan to achieve this 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 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 if one service is compromised others are still intact.

We have developed is 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.

We use Ubuntu as the base operating system. All the software is from Ubuntu <sup>1</sup> Edgy and subsequent distributions, which means that all used components are open source. We have only used existing software thus there will be little or no code to upkeep.

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

---

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

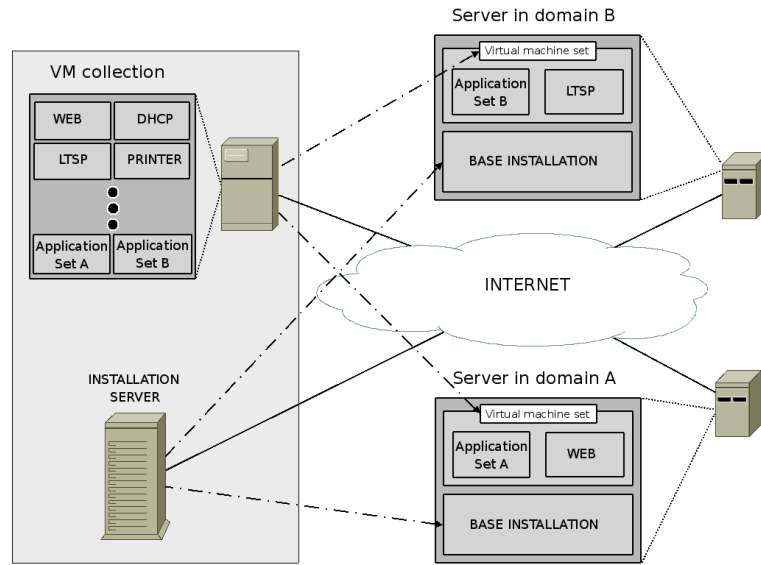


Figure 6.1: Prototype system overview

hardware virtualization but does not require it, which makes it suitable for various machines.

## 6.1 Operation of the system

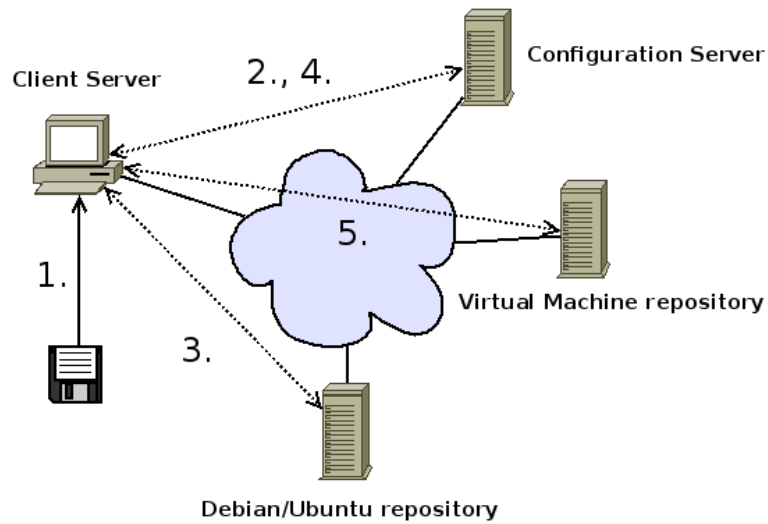


Figure 6.2: System's workflow

Workflow in short as illustrated in figure 6.1: 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.) Packagemanager is started and the packagelist is retrieved from the configuration server. 5.) Virtual machine installation packages retrieved from a repository and installed according to the list. 6.) Go back to step 4.

The client servers 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 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 to the client from the repository. All packages that were on the client and not on the new list are removed. The list can contain packages from the base operating system's repository and from a our own virtual machine repository.

Package lists contain all the software packages needed by the base installation. The lists also contains 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.

## 6.2 System components

### 6.2.1 Client installation media

Client installation media is used to install remote machines with a base operating system. Booting a machine with the installation media loads a 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.

Preseed file contains the answers to the questions that Debian Installer normally asks. All the configuration information on how to install the machine and where to get

installation packages etc. are included in the preseed file. A version of the preseed file, which is used in Finland, can be found from the appendix 8. Preseed files are stored and managed on a central server. Centrally located preseed files are easier to update and makes the actual installation media more generic.

### 6.2.2 Image server

The image server is basically a Debian/Ubuntu package repository i.e. a file server that can be accessed with HTTP. It is used to store Debian packages. The repository contains both the actual packages and the metadata of the packages in a predefined tree directory structure. The metadata is used by Debian and Ubuntu installation tools, the apt-tools, to install and manage packages and their dependencies [16].

The repository itself is created and managed with an open source tool called Reprepro<sup>1</sup>. Reprepro manages the versioning of packages and 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.

### 6.2.3 Image

Virtual machines are distributed by using image files. One image contains a filesystem 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 client machine. Scripts are used to prepare environment for the VM before it can be started 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<sup>1</sup>. It is also the base of Debian package management system. Dpkg is given as parameter a directory that contains all

---

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

<sup>1</sup><http://en.wikipedia.org/wiki/Dpkg>

the 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.

#### 6.2.4 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<sup>2</sup>. Rsync is a remote transfer program that keeps track of changes in files and eliminates all unnecessary transfers. The clients periodically synchronize their lists with the configuration server.

Usage of pull strategy was chosen to enhance the overall security of the system. The less incoming ports are the more insecure the system becomes [17]. Now the traffic is generated from inside and no extra holes are made for the incoming traffic.

#### 6.2.5 Packagemanager

PackageManager is an installation package management tool. It is installed as part of base installation. Packagemanager retrieves package lists from the configuration server and updates the machines software accordingly. It uses rsync and ssh 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. Pkgync 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.

### 6.3 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.

The library had old desktop PCs from which library the server was chosen. The server was installed using our systems. The library also obtained some economical, diskless

---

<sup>2</sup><http://samba.anu.edu.au/rsync/>

and powerwise limited computers <sup>3</sup>, which would nowadays be considered useless for anything else. But as thin clients they work well enough.

The server hosts two virtual machines. One 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 <sup>4</sup> desktop environment with all the required library software. For both virtual machines the applications were chosen carefully to minimize total system load. Communication between thin clients and the application server is done using X window system, which is by default very unsecure. 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.

---

<sup>3</sup>[http://www.icoptech.com/products\\_detail.asp?ProductID=271](http://www.icoptech.com/products_detail.asp?ProductID=271)

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

# Chapter 7

## Results

### 7.1 CERN library use-case

#### 7.1.1 Installation

#### 7.1.2 Management

#### 7.1.3 Comparison

### 7.2 Performance

#### 7.2.1 Image transfers

#### 7.2.2 Application memory usage

## Chapter 8

# Conclusions

Intrusion detection is like forecasting the weather...

# Bibliography

- [1] Myplc user's guide. <http://www.planet-lab.org/doc/myplc>. referred 2.12.2007.
- [2] Xen basics. <http://www.nodemaster.de/24-0-xen-basics.html>. referenced 22.11.2007.
- [3] *Configuring authentication and encryption*, January 2005.
- [4] *Understanding the Remote Desktop Protocol (RDP)*, March 2007. Article ID : 186607, Revision : 2.2.
- [5] 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.
- [6] 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.
- [7] H. K. Bjerke. Hpc virtualization with xen on itanium. Master's thesis, Norwegian University of Science and Technology, July 2005.
- [8] D. Blackman. Debian package management, part 1: A user's guide. *Linux J.*, 2000(80es):12, 2000.
- [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. referred 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>. referred 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. Smart-frog: 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] A. Kros and M. A. Margevicius. Thin-client shipments stage strongest growth since 2000. Technical report, Gartner, June 2006.
- [21] 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.
- [22] 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.
- [23] Microsoft. *Remote Desktop Protocol (RDP) Features and Performance*.
- [24] 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>. referred 26.09.2006, last edited 9.12.2005.
- [25] I. M.Revett and C.Stephens. Network computing: a tutorial review. *Electronics & Communications engineering journal*, February 2001.
- [26] A. Nye. *X Protocol reference Manual for X11 Version 4, Release 6*. O'Reilly & Associates, inc, 1995.
- [27] N. plc. Server based computing explained. <http://www.netvoyager.co.uk/general/sbce.html>. referred 21.8.2006, last edited.

- [28] T. Richardson. The rfb protocol. Technical report, RealVNC Ltd, 2007.
- [29] R. Rose. Survey of system virtualization techniques.
- [30] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.
- [31] A. Tijms. Binary translation: Classification of emulators. Technical report, Leiden Institute of Advanced Computer Science, 2000.
- [32] 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.
- [33] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble. Rethinking the design of virtual machine monitors. *Computer*, 38(5):57–62, 2005.
- [34] A. Whitaker and S. D. Gribble. propos of machine virtualization. [http://denali.cs.washington.edu/pubs/distpubs/slides/retreat\\_july\\_2001/VM\\_retreat\\_2.pdf](http://denali.cs.washington.edu/pubs/distpubs/slides/retreat_july_2001/VM_retreat_2.pdf).
- [35] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications, 2002.
- [36] S. J. I. D. P. T. Xavier Grehant, Olivier Pernet. Xen management with smartfrog.
- [37] 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.
- [38] A. Zeichick. Processor-based virtualization, amd64 style, part ii. [http://developer.amd.com/article\\\_print.jsp?id=15](http://developer.amd.com/article\_print.jsp?id=15). referred 3.11.2007, last edited 30.6.2006.

# Configuration files:

## .1 Preseed file

```
##### 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
```