UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Pós-Graduação em Ciência da Computação

Adriano Ricardo Digiere

Camada de Gerenciamento para Comunicação entre Computadores Baseada em Redes Sem Fio (WSE-OS)

UNESP
2011

Adriano Ricardo Digiere

Camada de Gerenciamento para Comunicação entre Computadores Baseada em Redes Sem Fio (WSE-OS)

Orientador: Profª. Adj. Roberta Spolon

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

UNESP

2011

ADRIANO RICARDO DIGIERE

Camada de Gerenciamento para Comunicação entre Computadores
Baseada em Redes Sem Fio (WSE-OS)

Dissertação apresentanda para obtenção do título de Mestre em Ciência da Computação, área de Sistemas de Computação junto ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

BANCA EXAMINADORA

Profª. Drª. Roberta Spolon
Professor Adjunto
UNESP – Bauru
Orientador

Prof. Dr. João Paulo Papa
Professor Assistente Doutor
UNESP - Bauru

Profª. Drª. Regina Helena Carlucci Santana
Professor Adjunto
USP/ICMC – São Carlos

São José do Rio Preto, 31 de Março de 2011

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, meu grande condutor nesta caminhada de vida. Agradeço por todas as vezes que me ajudou a encontrar o caminho a ser seguido, tanto na vida acadêmica quanto na vida pessoal.

Agradeço a minha orientadora Profª. Adj. Roberta Spolon, pela confiança em mim depositada desde o momento que me aceitou como orientando, pelas correções e adequações realizadas, pelas idéias que foram agregadas ao trabalho e por, diversas vezes, ter se disposto a ajudar e resolver problemas para que esse projeto pudesse ser desenvolvido.

Ao Prof. Adj. Marcos Antônio Cavenaghi, que acompanhou o desenvolvimento deste trabalho, ajudando sempre com sugestões, na organização das idéias, nas correções e submissões de artigos.

Aos meus queridos pais, José Roberto e Idelza, e meu irmão André Luis, por todo o esforço, dedicação e carinho que tiveram por mim durante toda a minha vida, possibilitando que eu sempre pudesse realizar meus sonhos.

Ao meu amigo Luis Gustavo Crepaldi, que compartilhou diversos momentos pesquisando, discutindo e aplicando o conhecimento obtido, proporcionando novas idéias e viabilidade na elaboração do projeto, além de sempre me ajudar em momentos de dificuldade técnica.

À UNESP, que proporcionou recursos para que os experimentos do projeto fossem concluídos.

À CAPES, pelo apoio financeiro.

À ABC71, por entender a importância do programa de mestrado, sempre me oferecendo suporte e liberdade para desenvolvimento do projeto.

Aos meus amigos, colegas e todos que colaboraram e me incentivaram na jornada para concretização deste trabalho, em especial à Profª. Adj. Renata Spolon Lobato e ao Prof. Dr. João Paulo Papa.

E por fim, à minha querida namorada e companheira Rebeca, por toda preocupação, incentivo e suporte, inclusive nos momentos em que estive ausente ou precisei dividir o tempo com o estudo.

“Imagination is more important than knowledge.”
Albert Einstein (1879 — 1955)

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3-DES: *Triple Data Encryption Standard*
ADSL: *Asymmetric Digital Subscriber Line*
AES: *Advanced Encryption Standard*
AMD: *Advanced Micro Devices*
AMD-V: *Advanced Micro Devices – Virtualization Technology*
ARM: *Advanced RISC Machine*
BIOS: *Basic Input/Output System*
BSD: *Berkeley Software Distribution*
CBC: *Cipher Block Chaining*
CD-ROM: *Compact Disc Read-Only Memory*
CPU: *Central Processing Unit*
DDR: *Double Data Rating*
DDR2: *Double Data Rating 2*
DDR3: *Double Data Rating 3*
DHCP: *Dynamic Host Configuration Protocol*
DMA: *Direct Memory Access*
DOS: *Disk Operating System*
ETRAX CRIS: *Ethernet, Token ring, AXis - Code Reduced Instruction Set*
EXT3: *Third Extended File System*
EXT4: *Fourth Extended File System*
FUSE: *Filesystem in USErspace*
GB: *Gigabyte*
GNOME: *GNU Network Object Model Environment*
GPL: *GNU General Public License*
GTK: *GIMP Toolkit*
GZIP: *GNU zip*
HVM: *Hosted Virtual Machines*
IBM: *International Business Machine*
IETF: *Internet Engineering Task Force*
IP: *Internet Protocol*
ISU: *Imagem de Sistema Única*
JIT: *Just-in-time*
JPEG: *Joint Photographic Experts Group*
KDE: *K Desktop Environment*
KVM: *Kernel-based Virtual Machine*
LGPL: *Lesser GNU General Public License*
LPS: *Loops per Second*
LTSP: *Linux Terminal Server Project*
MAC: *Medium Access Control*

MB/s: *Megabytes por Segundo*
MIPS: *Microprocessor without Interlocked Pipeline Stages*
MIPS: *Milhões de Instruções por Segundo*
MMU: *Memory Management Unit*
MPEG: *Moving Picture Experts Group*
MV: *Máquina Virtual*
NAT: *Network Address Translation*
OS2: *Operating System/2*
PNG: *Portable Network Graphics*
POSIX: *Portable Operating System Interface*
PSK: *Pre-Shared Key*
PUEL: *Personal Use and Educational License*
PV: *Para-Virtualizado*
PXE: *Pre eXecutable Environment*
RAM: *Random Access Memory*
RFB: *Remote FrameBuffer*
RMI: *Remote Method Invocation*
SCP: *Secure Channel Protocol*
SFTP: *SSH File Transfer Protocol*
SO: *Sistemas Operacional*
SOC: *Sistema Operacional Convidado*
SOH: *Sistema Operacional Hospedeiro*
SPARC: *Scalable Processor ARChitecture*
SSH: *Secure Shell*
SSHFS: *SSH FileSystem*
SSL: *Secure Sockets Layer*
TCP: *Transmission Control Protocol*
TCSC: *Thin Client Server Computing*
TI: *Tecnologia da Informação*
USB: *Universal Serial Bus*
UUID: *Unique Universal Identifier*
VDI: *Virtual Desktop Infrastructure*
VMM: *Virtual Machine Monitor*
VNC: *Virtual Network Computing*
VT-x: *Virtualization Technology, x86 Architecture*
WAV: *Waveform Audio File Format*
XDMCP: *X Display Manager Control Protocol*
XML: *eXtensible Markup Language*
WLAN: *Wireless Local Area Network*
WPA2: *Wi-Fi Protected Access 2*
WSE-OS: *Wireless Sharing Environment – Operating Systems*

RESUMO

O maior custo de propriedade de computadores não é o *hardware* ou o *software*, mas sim o tempo que os profissionais de informática gastam em suporte e manutenção dos ambientes computacionais. Em um conglomerado de computadores em rede, cada computador torna-se uma entidade gerenciada individualmente, o que gera contínuas solicitações de alterações de configuração, como instalação de atualizações de *software*, conexão e configuração de periféricos, criação de perfis de *e-mail* e aplicação de *patches*. Além disso, existe ainda o risco de furto de dados e invasão por *hackers* quando os computadores dos usuários não estão protegidos. Aliado a este cenário, a constante evolução dos sistemas computacionais e seu potencial de processamento, a cada dia são necessárias novas técnicas de aproveitamento destes recursos. Soluções que visam facilitar o gerenciamento de ambientes com grande massa de computadores de forma a tirar o máximo proveito do poder computacional concentrado em servidores já se tornaram necessidades reais, não só em grandes corporações, mas também em pequenas e médias empresas, além de outros tipos organizações, como por exemplo, instituições de ensino. Frente esta necessidade, focando uma ferramenta compatível neste cenário de crescimento, este trabalho apresenta um modelo de gerenciamento centralizado, nomeado WSE-OS (*Wireless Sharing Environment – Operating Systems*), baseado em técnicas de virtualização e acesso remoto seguro combinadas a um sistema de arquivos remotos em espaço de usuário. Esta solução elimina a necessidade da instalação e configuração de aplicativos "máquina a máquina", além de tirar maior proveito do poder computacional existente nos servidores. A principal característica deste modelo que o destaca das soluções atuais é que ele é especificamente elaborado para operar sobre redes com baixas taxas de transmissão, como as redes sem fio. O WSE-OS é capaz de realizar a replicação de imagens de sistemas operacionais em um ambiente com comunicação WLAN, tornando o gerenciamento mais flexível e independente de conexões físicas, além de oferecer abstração de *hardware* clientes para melhor desempenho. Durante o trabalho foram desenvolvidos os módulos localizados no servidor WSE-OS que compõem parte do modelo proposto e que confirmam se tratar de uma solução robusta, segura e escalável de VDI (*Virtual Desktop Infrastructure*). Os demais módulos integrantes, que estão presentes nos clientes, são desenvolvidos por outro trabalho. Os resultados obtidos indicam que o WSE-OS é capaz de disseminar, através de uma única configuração de *software*, a execução de dados relativos à imagem de sistemas operacionais em computadores clientes, podendo ser aplicado como ferramenta de gerenciamento em ambiente de compartilhamento sem fio para sistemas operacionais.

PALAVRAS-CHAVE: *Virtual Desktop Infrastructure*; Gerenciamento de Computadores; Máquinas Virtuais; Acesso Remoto; Sistema de Arquivos Remotos.

ABSTRACT

The largest cost of desktop ownership is not the hardware or software, but the time that administrators spend on support and maintenance of computing environments. In a conglomerate of computers in a network, each computer becomes an entity managed individually, which generates continuous requests for configuration changes, such as installing software updates, configuration and connection of peripherals, profiling email and applying patches. Moreover, there is the risk of data theft and hacking when users' computers are not protected. Allied to this scenario, the constant evolution of computer systems and their potential for processing, each day requires new techniques for exploitation of these resources. Solutions aimed facilitating the management of environments with large mass of computers to take maximum advantage of computing power concentrated on servers have become real needs, not only in large corporations but also small and medium enterprises, besides other types organizations, such as educational institutions. Facing this need, focusing on a tool that supported this growth scenario, this work presents a centralized management model, named WSE-OS (Wireless Sharing Environment – Operating Systems) based on virtualization techniques and secure remote access combined with a remote file system in user space. This solution eliminates the need for installing and configuring applications "machine to machine", besides take greater advantage of existing computing power on the servers . The main feature of this model that highlights the current solutions is that it is specifically designed to operate on networks with low transmission rates, such as wireless networks. The WSE-OS is able to perform the replication of operating system images in an environment with WLAN communication, which makes management more flexible and independent of physical connections, besides offer customer's hardware abstraction for better performance. During the study, were developed modules located on the server OS-WSE that comprise part of the proposed model and confirm that it is a robust, secure and scalable VDI (Virtual Desktop Infrastructure) solution. The remaining members modules, which are present in customers, are developed by another study. The obtained results indicate that the WSE-OS is capable of disseminating, via a single software configuration, execution of data related to the image of operating systems on client computers, and can be applied as a management tool in wireless sharing environment for operating systems.

KEYWORDS: Virtual Desktop Infrastructure, Computer Management, Virtual Machines, Remote Access, Remote File System.

1. INTRODUÇÃO

Esta seção apresenta o escopo e a motivação para o trabalho proposto neste documento. A seção 1.1 mostra os contextos nos quais o trabalho está inserido, a seção 1.2 mostra as justificativas para a elaboração do projeto. Na seção 1.3 são apresentados os objetivos gerais do trabalho e, por fim, a seção 1.4 detalha a estrutura desta monografia.

1.1 Escopo de Trabalho

A presença de computadores nas organizações surge em função do seu impacto no aumento de produtividade e capacidade de gerar valor para empresas. Em paralelo ao aumento da presença dos computadores nas organizações, veio o surgimento e gradativo aumento de demanda para ferramentas de gerenciamento de redes de computadores capazes de facilitar a tarefa dos gerenciadores de ambientes computacionais. A atual tecnologia de redes locais de alta velocidade e processadores de baixo custo vêm colaborando para que organizações adotem soluções de gerenciamento centralizado de computadores, como as soluções *thin client*[1], que exploram a arquitetura cliente-servidor (BARATTO, KIM & NIEH, 2005). Até mesmo em ambientes educacionais, como laboratórios de informática com computadores em rede, pode-se ter um enorme benefício através de uma solução planejada de gerenciamento de computadores para obtenção de um nível dinâmico de maturidade computacional (MORGADO, CRUZ & TWANI, 2008), onde as ferramentas de provisão e gerenciamento de configuração de *software* contribuem para a estabilidade e padronização do ambiente de execução (CRUZ, 2008).

O gerenciamento de um ambiente computacional dotado de um conglomerado de dispositivos em rede é uma atividade potencialmente complexa em função da possível natureza heterogênea desses equipamentos. Esses ambientes podem apresentar computadores com configurações diversas de *software* básico e aplicativos em função das diferentes configurações de *hardware* em cada nó da rede. Neste cenário, cada computador

[1] Modelo de gerenciamento centralizado em que os computadores clientes em uma rede de arquitetura cliente-servidor possuem poucos ou nenhuns aplicativos instalados, de modo que dependem primariamente de um servidor central para o processamento de atividades.

precisa ser gerenciado individualmente, refletindo em uma atividade manual de configuração da imagem de sistema ou com automatização limitada à camada de aplicativos (CRUZ, 2008).

Frente esse impasse, algumas soluções de gerenciamento centralizado de computadores em redes contornam este forte acoplamento entre *hardware* e *software* através de tecnologias de virtualização. Neste contexto, a virtualização é utilizada como ferramenta de otimização.

Em conjunto com a virtualização, as redes podem contribuir com o gerenciamento centralizado de computadores, pois possibilitam tanto a propagação de instruções a diversas máquinas quanto um acesso centralizado a um único computador que pode possuir os sistemas utilizados pelo ambiente.

Algumas ferramentas de gerenciamento centralizado, como o *Citrix Provisioning Server for Desktops*[2] e o *Ardence Desktop Streaming*[3], propõem uma solução que combina virtualização do disco rígido das estações com mecanismo de *boot* remoto. Neste caso, somente o disco rígido é virtualizado por meio de uma Imagem de Sistema Única (ISU) que é montada através da rede como um dispositivo de acesso a blocos pelas estações clientes, em tempo de *boot*, como se fosse um disco rígido do próprio cliente (CRUZ, 2008). Através do protocolo PXE (*Preboot eXecution Environment*) (INTEL, 1999) os computadores realizam o *boot* utilizando remotamente uma imagem de disco rígido, como se este estivesse no próprio computador. Esse tipo de solução oferece um ótimo compromisso entre desempenho e requisitos de *hardware* oferecendo uma grande capacidade de escala, porém, sofre com o forte acoplamento entre *hardware* e *software*, o que pode exigir que cada computador de um ambiente heterogêneo de *hardware* tenha a sua própria ISU (CRUZ, 2008).

Outras soluções de mercado, como as baseadas em VDI (*Virtual Desktop Infra-structures*), utilizam a virtualização para criar uma infra-estrutura virtual de *desktops* (VMWARE, 2007), em uma arquitetura similar a arquitetura *thin client*. Neste caso a virtualização é utilizada para criar *desktops* virtuais no servidor de virtualização e esses *desktops* são acessados por meio de protocolos de conexão pelos computadores da rede. As limitações impostas por

[2] http://www.citrix.com.br/products/provisioning_server_desktops.php
[3] http://www.ardence.com

este tipo de solução ficam por conta da centralização do processamento e pela impropriedade para aplicações com altos requisitos de rede, como por exemplo, aplicações multimídias.

Como outra ferramenta para gerenciamento centralizado, tem-se o FLEXLab (AGUIAR et al., 2009), uma solução que combina técnicas de virtualização com a utilização de sistemas de arquivos distribuídos utilizados em *clusters* de computadores de modo a permitir a criação de uma solução de gerenciamento centralizado com processamento distribuído nos computadores da rede, ao contrário de uma solução *thin client*.

Entretanto, grande parte das atuais soluções existentes no mercado para gestão centralizada, penaliza a mobilidade, a ubiquidade e poder computacional. Pelo fato destes ambientes de compartilhamento serem estruturados sobre rede cabeada, os mesmos acabam limitando a flexibilidade e facilidade de instalação. Em contraste a esse cenário, nos últimos anos os equipamentos de redes sem fio vêm melhorando no aspecto tecnológico, tornando-se uma tecnologia extremamente popular.

Já as soluções que priorizam a flexibilidade e ubiquidade, como as baseadas em VDI, muitas vezes tornam inapropriada a utilização de determinados aplicativos com altos requerimentos de rede, contrastando com o cenário atual, onde os aplicativos vêem, cada vez mais, se tornando multimídias e interativos.

Diante deste cenário de limitação, é possível, através da combinação das técnicas adequadas de virtualização com a utilização de sistemas de acesso remoto eficientes, a criação de uma solução de gerenciamento centralizada sem fio, ao contrário das soluções convencionais cabeadas, e com uma comunicação de rede capaz de suportar aplicações que requerem altas taxas de transmissão. Esta abordagem representa uma nova perspectiva em soluções para gerenciamento de ambientes de compartilhamento de sistemas operacionais e propõe uma nova utilização para a tecnologia de acesso remoto.

1.2 Motivação

As redes sem fio vêm cada vez mais ganhando espaço na comunicação entre dispositivos dos

mais variados tipos e tamanhos, desde computadores pessoais, passando por celulares, até eletrodomésticos e máquinas industriais, e nos mais diversos ambientes, como residências, escritórios, empresas e bibliotecas. Pelo fato de o meio de transmissão utilizado pelas redes sem fio ser não-delimitado, no caso a atmosfera, onde os sinais provenientes do transmissor se propagam livremente no meio e se difundem através dele, estas redes contribuem para a ubiquidade do poder computacional, tornando transparente a disseminação das informações e a cooperação dos dispositivos na realização das mais variadas tarefas.

Devido ao rápido avanço tecnológico, os equipamentos necessários para configuração de uma rede sem fio, vêm se tornando acessíveis a custos cada vez mais reduzidos, fazendo com que tecnologias sem fio passem cada dia mais a adquirir grandes fatias no mercado consumidor. Além disto, tendências recentes da indústria computacional, como a consolidação de servidores e ambientes tolerantes a falhas, associados à redução do custo de processadores e memórias, o surgimento dos processadores de múltiplos núcleos para computadores pessoais e a necessidade de aumentar a eficiência operacional de *data-centers* têm contribuindo para o ressurgimento do interesse nas tecnologias de virtualização de *hardware*, especialmente nas máquinas virtuais (ROSENBLUM, 2004). Dentre as inúmeras possibilidades que surgem em função do uso de virtualização de *hardware*, o melhor aproveitamento do poder computacional ocioso dos equipamentos é apenas uma das vantagens fornecidas por esta tecnologia (SMITH & NAIR, 2005).

Resumidamente, a virtualização oferece a possibilidade de executar múltiplos Sistemas Operacionais (SOs) em uma única plataforma física, dividindo todos os recursos de *hardware* entre as diversas instâncias de SO em execução (BARHAM et al., 2003). Em outras palavras, a técnica de virtualização permite com que dois "computadores virtuais", com SOs e aplicações diferentes, executem simultaneamente sobre um único computador real de forma isolada, ou seja, a execução de um SO não interfere na de outro. Desta forma, a virtualização pode ser utilizada com o objetivo de aumentar os serviços de um ambiente de TI[4], com menos manutenções físicas, já que centraliza toda e qualquer manutenção em um único equipamento.

[4] A Tecnologia da Informação (TI) é o conjunto de todas as atividades e soluções providas por recursos de computação.

Outro sistema de grande auxílio em gerenciamento de redes são os sistemas de acesso remoto. O acesso remoto permite a comunicação entre duas máquinas, de tal forma a viabilizar a troca de informações entre as aplicações (REALVNC, 2009), simplificando a complexidade de relacionamento em uma organização ao fornecer a possibilidade de interação em tempo real entre recursos distantes.

Diante destas características peculiares às máquinas virtuais, associada aos sistemas de acesso remoto e as redes sem fio, é possível a criação de um modelo para gerenciamento de computadores, onde uma configuração de *software* é capaz de atender os computadores em uma rede sem fio de um determinado ambiente, simplificando o gerenciamento e a disponibilização destes computadores.

1.3 Objetivos Gerais

O objetivo deste trabalho é apresentar um modelo de arquitetura distribuída para ambientes computacionais, denominado WSE-OS (*Wireless Sharing Enviroment – Operating Systems*), que permite o compartilhamento de sistemas operacionais por diversos usuários através de rede sem fio. Tal modelo visa proporcionar maior facilidade para a tarefa de gerenciamento de ambientes computacionais, centralizando as configurações da camada de *software* em um ponto central, desta forma evitando o exaustivo trabalho de configuração "máquina a máquina" do ambiente.

O modelo de arquitetura distribuída proposto possui duas entidades independentemente estruturadas, dispostas separadamente entre clientes e servidor WSE-OS que, quando combinadas, são capazes de configurar um sistema de compartilhamento para instanciação e execução de sistemas operacionais em ambientes sem fio. Desta forma, o WSE-OS também serviu de base para outro projeto de pesquisa concluído que estudou e avaliou a aplicação de um *middleware* (mediador) para execução em terminais clientes como módulo integrante do projeto em discussão.

Diante desta estruturação bilateral, esta dissertação tem por objetivos específicos:

- Desenvolvimento de uma camada de gerenciamento para execução em servidores de

arquitetura x86[5] e x64[6] permitindo o compartilhamento de execução de imagens de sistemas operacionais (localizadas no servidor) por diversos usuários;

- Incorporação à camada de gerenciamento de um sistema seguro de arquivos remotos fornecendo aos clientes a possibilidade de acesso a seus dispositivos removíveis;
- Desenvolvimento de uma interface gráfica com os usuários do sistema para permitir escolha de qual sistema operacional, da lista disponível, será instanciado;
- Realização de experimentos para analisar o desempenho da arquitetura da camada de gerenciamento, bem como de ferramentas de virtualização consideradas para o modelo proposto a fim de justificar a escolha feita por este trabalho.

1.4 Estrutura da Monografia

O trabalho está organizado em sete capítulos. Neste primeiro capítulo foi apresentada a contextualização do trabalho, as motivações e os objetivos que impulsionaram o desenvolvimento do mesmo.

O Capítulo 2 apresenta conceitos sobre ambientes de trabalho remoto abordando as arquiteturas das soluções mais conhecidas. Além disso, ressalta as principais vantagens e desvantagens de cada uma das soluções, apresentando, ao final, um comparativo entre elas.

O Capítulo 3 tem por objetivo apresentar a tecnologia de virtualização e como foram inicialmente concebidas, descrevendo as máquinas virtuais, com suas classificações e tipos. Ainda no capítulo 3, são abordadas as técnicas de virtualização, os desafios que a arquitetura x86 apresenta à virtualização clássica, paravirtualização e demais técnicas. Neste capítulo são apresentados também exemplos de máquinas virtuais.

O Capítulo 4 apresenta o modelo WSE-OS associado com características técnicas e fluxo de execução do modelo arquitetural, além dos componentes fundamentais e objetivos específicos no qual esta dissertação está focada.

[5] Nome genérico dada à família (arquitetura) de processadores baseados no Intel 8086.
[6] Nome genérico dada à família (arquitetura) de processadores baseados na tecnologia de 64 *bits*.

O Capítulo 5 descreve os módulos que compõem a arquitetura da camada de gerenciamento WSE-OS bem como características técnicas e funcionamento para que possa ser aplicado como parte integrante do projeto em questão.

O Capítulo 6 apresenta *benchmarks* realizados sobre as soluções de virtualização consideradas para o modelo proposto, discutindo os resultados obtidos. Posteriormente é apresentado um teste sobre a Camada de Gerenciamento WSE-OS a fim de verificar o desempenho da solução em função do número de usuários no ambiente.

A conclusão deste trabalho encontra-se no Capítulo 7, que mostra como os objetivos foram alcançados. São apresentadas as contribuições do projeto, as limitações atuais da solução e as propostas para trabalhos futuros que podem colaborar para o modelo.

2. AMBIENTES DE TRABALHO REMOTO

2.1 Introdução

Nos dias atuais, com o crescente aumento do desempenho de comunicação por redes e da capacidade de processamento dos equipamentos computacionais, torna-se cada vez mais viável, principalmente em grandes empresas, o uso de Servidores de Terminais (ou Servidores de Aplicações), também chamados de servidores de interfaces gráficas. Através desses servidores dotados de sistemas operacionais multiusuário, é possível a abertura de ambientes gráficos por clientes remotos. Desta forma, fica separado o sistema operacional nativo (aquele que o usuário possui em seu computador) do sistema operacional visualizado (aquele que está sendo acessado remotamente pelo cliente).

Os clientes funcionam como terminais “burros”, tendo acesso somente às aplicações existentes no servidor. Todo o processamento do cliente é realizado no segundo, que transmite apenas a interface do ambiente para o cliente. Já este devolve movimentos e cliques do *mouse* e teclado para serem processados pelo servidor.

Para o usuário, a impressão que se tem é de que o sistema operacional remoto é executado localmente. Evidentemente, existem algumas limitações, como por exemplo, o acesso, por parte do cliente, aos dispositivos da máquina remota, sendo necessária a montagem remota dos mesmos. Outra dificuldade enfrentada quando se tem acesso a terminais remotos, é a reprodução de áudio localmente de mídias executadas remotamente.

O uso de Servidores de Terminais em ambientes corporativos e acadêmicos facilita o gerenciamento da TI, possibilitando o controle de privilégios dos usuários e diminuindo a chance de perda de dados por erros dos próprios usuários ou por problemas relacionados a *hardware*. Além disso, centralizar o SO facilita a monitoração do sistema e diminui os gastos com *hardware*, pois é mais fácil monitorar e mais barato investir em uma única máquina.

Existem diversos aplicativos que permitem o acesso ao ambiente gráfico de um servidor

remoto. Uma das soluções mais antigas e que se difundiu é o VNC[7] (*Virtual Network Computing*). Ele permite que um usuário utilizando um computador dotado de qualquer um dos mais conhecidos sistemas operacionais (*Linux*, *Windows*, *MacOS*, *Solaris*, *Amiga*) faça uso de um ambiente gráfico de um servidor, podendo também este possuir qualquer um dos mesmos sistemas operacionais (não necessariamente o mesmo do cliente). Tal característica foi responsável por sua enorme difusão, mesmo o VNC não apresentando qualidade de imagem e desempenho satisfatórios.

Outra solução, também antiga, é o XDMCP (CHAO, 2003), protocolo nativo do *X Window System* (COOPERSMITH, 2010), ou simplesmente X, servidor gráfico presente em praticamente todos os sistemas Linux. O X oferece uma qualidade melhor de imagem que o VNC, porém gera um tráfego de rede proporcionalmente maior, inviabilizando seu uso fora de redes locais.

Em 2003, a empresa NoMachine apresentou o NX (REGIS, 2007), uma solução que supera todas as demais em praticamente todos os âmbitos. O NX utiliza o SSH (*Secure Shell*) (YLONEN, 2006a) para encriptar os dados, proporcionando maior segurança à troca de dados, e os protocolos de compressão JPG[8] e GZIP[9], aumentando a velocidade do tráfego de dados. O NX também apresenta mais ferramentas que os demais. Tais ferramentas são apresentadas na seção 2.4.

Recentemente, a Google anunciou sua própria aplicação cliente-servidor para telas remotas, com algumas bases no NX. O projeto é nomeado de *NeatX* (GOOGLE, 2010), e tem por objetivo se transformar em uma solução simples e rápida, além de aberta. O projeto é recente, anunciado em Julho de 2009, e ainda tem muito a ser desenvolvido, inclusive a Google vem aceitando contribuições de voluntários. A seção 2.5 apresenta esta recente tecnologia lançada pela Google.

[7] http://www.realvnc.com/support/documentation.html
[8] JPG (*Joint Photographic Experts Group*) é um método comumente usado para comprimir imagens fotográficas.
[9] Abreviação de GNU zip, consiste em um *software* livre de compressão sem perda de dados.

2.2 *X Window System*

2.2.1 Descrição

O *X Window System*, ou simplesmente X, consiste em um protocolo que visa acesso a interfaces gráficas, localmente ou através da rede, em sistemas operacionais baseados em *Unix*.

O X surgiu como resultado de pesquisas em construção de aplicações multiprocessadas distribuídas, e foi desenvolvido para ser um padrão, ou seja, uma especificação de *software*, para possibilitar a construção de aplicações gráficas interativas.

2.2.2 Histórico

O X surgiu em 1984 no *Massachusetts Institute of Technology* (MIT) como parte de pesquisas relacionadas com o Projeto Athena e o Sistema Argus[10], ambos voltados ao desenvolvimento de aplicações multiprocessadas distribuídas. Ele foi criado para ser usado em *mainframes* rodando *Unix*, usados em conjunto com estações de trabalho que se limitavam a apresentar as imagens na tela dos aplicativos executados no servidor.

Em 1993, o X foi portado para o *Linux* e rapidamente se tornou o servidor gráfico mais utilizado nesta plataforma. Ao longo de sua história, o X possuiu duas implementações principais: *XFree86*[11] e *X.org*[12] (uma bifurcação do *FreeX86*). Até 2004, o *XFree86* foi a implementação mais utilizada do X, porém a partir daí o *X.org* tem predominado. Tal mudança de cenário se deve à problemas relacionados à licença do *XFree86*.

Atualmente, o X é desenvolvido e distribuído pelo X Consortium, uma associação formada por um grande número de fabricantes de computadores. O X passou a ser bastante disseminado a partir de sua versão 11, também conhecida como X11, e hoje se encontra na

---

[10] http://web.mit.edu/annakot/MacData/afs/athena.mit.edu/astaff/project/argus/doc/functional.spec/history.mss

[11] http://www.xfree86.org/

[12] http://www.x.org/wiki/

versão X11R7 (versão 11, revisão 7) (COOPERSMITH, 2010).

2.2.3 Funcionamento

O X utiliza o modelo cliente-servidor. O servidor é responsável por cuidar das requisições aos dispositivos gráficos (monitor, impressora, etc.) enquanto o cliente se comunica com o servidor fazendo requisições do tipo "desenhe uma linha" ou "atenção para a entrada de dados do teclado" e recebe do servidor os eventos de entrada (teclado e mouse) (GUEDES & SILVA, 2006). Servidor e cliente podem estar na mesma máquina ou em máquinas diferentes, possibilitando o acesso a servidores remotos de ambientes gráficos.

A Figura 1 mostra um modelo cliente-servidor do X. Neste modelo, o cliente X e o servidor X estão em máquinas independentes, que estão interligadas através de uma rede, mas poderiam estar, eventualmente, na mesma máquina.

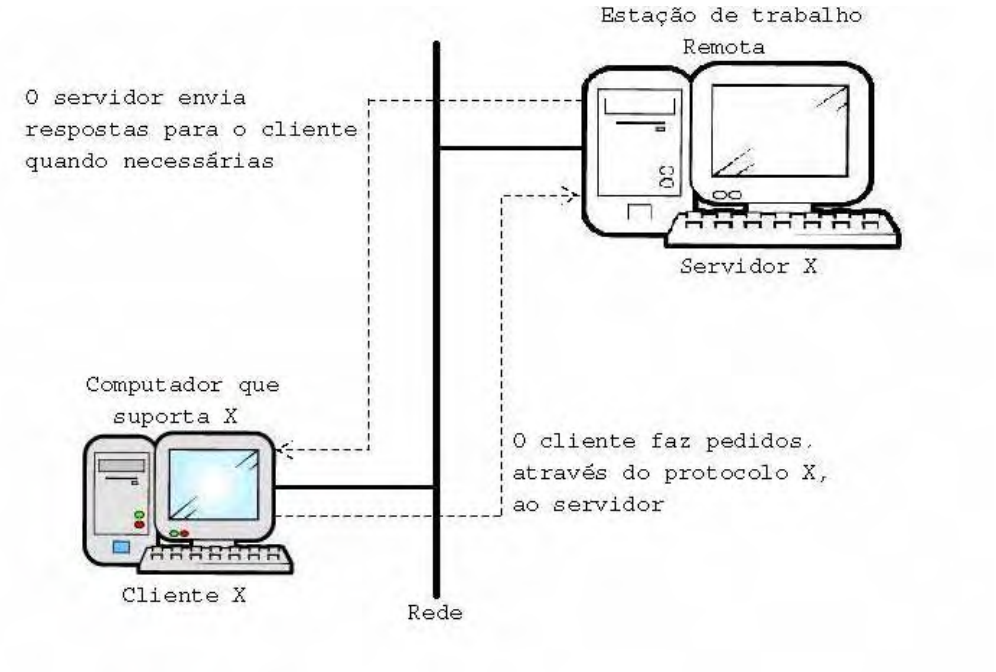

Figura 1 - Modelo cliente servidor do *X Window System.*

O X não deve ser visto como uma interface de usuário. Ele fornece mecanismos para que diferentes tipos de interfaces possam ser executados sobre ele. O gerenciamento das interfaces é função dos clientes e bibliotecas externos ao servidor X e às próprias bibliotecas

do sistema (SILVA, 2000). Tais bibliotecas são descritas na subseção seguinte.

2.2.4 Arquitetura

O X possui um protocolo nativo de compartilhamento de *desktops*, que possibilita o acesso remoto e *thin client* (ARAUJO, 2006). Ele foi desenvolvido na década de 80 para ser usado em mainframes em conjunto com terminais que tinham por finalidade apenas apresentar na tela as imagens dos programas executados no servidor. Na década de 80, o custo de *hardware* necessário para suportar aplicativos gráficos era muito elevado, por isso compartilhar um servidor.

Em função desse cenário, o X foi desenvolvido em cima de um protocolo rápido e robusto de comunicação via rede. As imagens são transmitidas na forma de comandos, consumindo pouca banda de rede, e são processadas pelos destinatários de forma bem rápida (MORIMOTO, 2009a). Isto possibilita que o ambiente remoto seja visualizado pelo usuário sem atrasos.

O modelo de computação utilizada pelo X é o modelo cliente-servidor, ou seja, um servidor pode se comunicar com diversos clientes. Além disso, o X permite a exibição de janelas tanto do sistema local, quanto de outros servidores X remotos. Isto permite que clientes possam visualizar um ambiente gráfico instanciado em um servidor X remoto.

A forma de comunicação entre cliente e servidor se dá por trocas de mensagens (pacotes) via *socket* na rede. A conexão é estabelecida pelo cliente. Este envia um pacote requisitando uma conexão ao servidor, que pode ou não aceitá-la. Há ainda um terceiro caso, no qual o servidor pode exigir uma confirmação de acesso (controle de acesso).

Quando estabelecida a conexão, quatro tipos de pacotes podem ser trocados via rede (OLIVEIRA et al., 2006):

- Pacote de requisição (*Request*): através deste pacote, o cliente solicita informações ao servidor;
- Pacote de resposta (*Reply*): é através deste pacote que o servidor responde às requisições do cliente, porém nem todas as solicitações precisam ser respondidas;

- Pacote de evento (*Event*): pacote enviado pelo servidor para comunicar um evento ao cliente, tal como entradas provenientes do teclado ou mouse, movimento de uma janela, etc.;
- Pacote de erro (*Error*): pacote enviado pelo servidor para informar ao cliente de que uma requisição foi inválida.

O mecanismo de troca de mensagens entre cliente e servidor X pode ser entendida pela Figura 2:

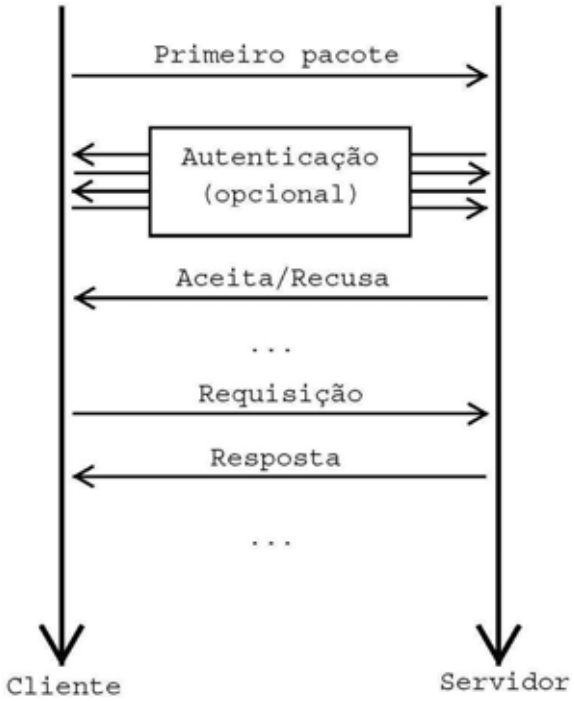

Figura 2 - Troca de mensagens entre cliente X e servidor X. Adaptado de (OLIVEIRA et al., 2006).

Existem alguns elementos fundamentais para que se dê a comunicação entre cliente e servidor X que são de responsabilidade do segundo. Tais elementos são apresentados nas subseções seguintes.

2.2.4.1 Janela

Uma janela consiste em uma interface gráfica que pode ser composta por alguns componentes como botões, menus, ícones, imagens, barras de rolagem, *menu*, dentre

outros.

Uma janela apenas pode ser criada a partir de outra janela já existente, o que faz do sistema hierárquico, onde se tem janelas filhas de outras. A janela principal, conhecida como "janela *root*", é criada pelo servidor de forma automática e consiste em uma janela em tela cheia ("*full screen*") que fica "atrás" de todas as demais janelas (OLIVEIRA et al., 2006).

2.2.4.2 Identificadores

Todos os dados das janelas são armazenados no servidor. Para que o cliente possa acessar tais objetos, faz-se necessário o uso de um elemento que permita a identificação de tal janela tanto para o cliente quanto para o servidor. Tal elemento é conhecido como identificador (OLIVEIRA et al., 2006).

Quando um cliente deseja criar uma janela, ele faz uma requisição junto ao servidor para que ela seja criada, passando um identificador ao segundo. O servidor, por sua vez, cria a janela associando-a ao identificador recebido do cliente. Tal identificador permite que o cliente possa, posteriormente, alterar tal janela, como por exemplo, escrever uma frase nela (OLIVEIRA et al., 2006).

O identificador é único no servidor, ou seja, sempre que um cliente deseja criar uma nova janela, deve-se associá-la a um novo identificador (OLIVEIRA et al., 2006).

2.2.4.3 Atributos e propriedades

Todas as janelas possuem informações que devem ser armazenadas pelo servidor e que podem ser acessadas pelos clientes através de requisições. Tais informações podem ser divididas em duas classes: atributos e propriedades.

Atributos são características associadas à janela, tais como tamanho, posição, cor de fundo e outros. Propriedades são valores dos dados anexados às janelas. Ao contrário dos atributos, as propriedades não são pré-identificadas pelo servidor (OLIVEIRA et al., 2006).

São através das propriedades que os clientes podem armazenar dados associados à janela e que são relevantes a ele. A cada propriedade é associado um nome, um tipo e um valor. Elas são similares às variáveis, podendo o cliente criar uma nova e armazenar valores (OLIVEIRA et al., 2006).

2.2.4.4 Eventos

Eventos são pacotes de dados enviados pelo servidor ao cliente para comunicar a este de que alguma coisa ocorreu. Como por exemplo, quando o segundo faz uma requisição ao primeiro para saber qual parte do texto está selecionado no momento, este envia um evento ao cliente que está manipulando a janela no momento (OLIVEIRA et al., 2006).

Existem tipos de eventos que são enviados continuamente ao cliente, porém a maioria deles é enviada somente quando este explicita que necessita deles (OLIVEIRA et al., 2006).

2.2.4.5 Bibliotecas

Existem, no X, duas bibliotecas que facilitam e simplificam o desenvolvimento de aplicações gráficas, oferecendo tipos de dados e rotinas para permitir a interação com o sistema de janelas. Essas duas bibliotecas são conhecidas como *Xlib* e *X Toolkit*.

A biblioteca *Xlib* possui funções gráficas primitivas que permitem a criação de interfaces de baixo nível, enquanto a biblioteca *X Toolkit*, construída "sobre" a *Xlib*, provê funções que tratam dos elementos que constituem a interface, os *widgets*[13], obtendo assim, os mecanismos necessários para a construção de interfaces em alto nível (SILVA, 2000).

A infra-estrutura do *X Toolkit* é uma biblioteca implementada em linguagem C, conhecida como *libXt*. Sua principal utilidade se dá quando combinada com uma biblioteca de *widgets* pré-construídos (*widget set*) com menus, botões de comando e barras de rolagem, originando assim uma aplicação com "*look and feel*" consistentes. Com o uso dos *widgets*, os

[13] Um widget é um componente de interface gráfica com o usuário (GUI). Qualquer item de uma interface gráfica é chamada de *widget*, por exemplo: janelas, botões, menus e itens de menus, ícones, barras de rolagem, etc

códigos de aplicação e da interface ficam separados (SILVA, 2000).

O *widget* é independente da aplicação, porém ela pode alterar os atributos do *widget* através de invocações de funções do *toolkit*.

2.2.4.6 Gerenciadores de janela

Um conceito relacionado ao X é o de gerenciadores de janelas. Um gerenciador de janelas é um programa que funciona junto com o servidor X e é ele que controla a aparência das janelas e elementos gráficos. É o gerenciador de janelas que dita como o usuário vai interagir com a interface gráfica. Seu papel principal é prover uma interface amigável e padronizada ao usuário, fornecendo também aplicativos e utilitários que possam ser úteis (OLIVEIRA et al., 2006).

Como exemplos de gerenciadores de janelas, pode-se citar o KDE (KDE, 2010), GNOME (THE GNOME PROJECT, 2010) e *WindowMaker* (KOJIMA, 2011).

2.2.5 Características do *X Window System*

Como pontos positivos do X, tem-se que ele possui uma arquitetura altamente escalável, suporte a extensões e módulos e usa *drivers* nativos. Além disso, não se deve esquecer de seus 25 anos de desenvolvimento e que ele suporta o VNC e o NX, dando ainda mais credibilidade ao sistema.

Uma das principais desvantagens do X são os "*roundtrips*" (pacotes de resposta). O X foi desenvolvido para funcionar localmente (na mesma máquina) ou em rede local, onde o tempo de latência do *link* é relativamente muito baixo (MORIMOTO, 2006). Por ter sido desenvolvido visando esse cenário, o X envia um pacote de resposta para cada instrução recebida, garantindo assim integridade da transmissão. O servidor X apenas transmite a próxima instrução se receber resposta para a última instrução enviada. Tomando como exemplo um aplicativo como o *Firefox*, pode-se ter a necessidade da transmissão de mais de 2000 pacotes de resposta durante seu carregamento inicial (MORIMOTO, 2006). A Figura 3

ilustra este cenário de “*roundtrips*”.

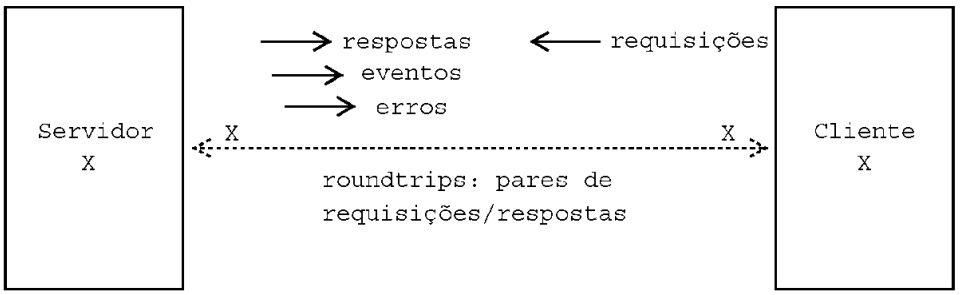

Figura 3 - Os "*roundtrips*" do X. Adaptado de (REGIS, 2007).

Em função disso, tem-se que o X não apresenta um bom desempenho para aplicações que apresentam interações rápidas, como por exemplo, jogos.

2.3 VNC

2.3.1 Descrição

O VNC (*Virtual Network Computer*) consiste em um aplicativo de acesso remoto open source que disponibiliza a um usuário não apenas dados e aplicativos, mas um ambiente *desktop* completo através da internet ou rede local (MORIMOTO, 2006).

Apesar de ser um dos aplicativos de acesso remoto mais antigos, ele continua sendo utilizado na comunidade computacional devido à sua facilidade de uso e sua flexibilidade. Ele não oferece um sistema de compressão otimizado, no entanto, ele é simples de usar e tão flexível que pode ser usado até mesmo através de um navegador *web*.

Existem versões do VNC para a maioria das plataformas: *Linux*, *Windows*, *MacOS*, *Solaris*, *Amiga* e atualmente pode-se ter o VNC até mesmo em *Windows Mobile* e *Palm OS*, possibilitando sua presença em dispositivos móveis.

2.3.2 Histórico

O VNC foi desenvolvido pela AT&T (*American Telephone and Telegraph*) *Corporation* como uma ferramenta para possibilitar interfaces gráficas remotas e rapidamente substituiu os mais utilizados programas de acesso remoto da década de 90. O *software* atualmente mantido pelos autores do VNC chama-se *RealVNC* e pode ser encontrado no site[14] da própria empresa.

As duas versões mais utilizadas do VNC são: *RealVNC* e *TightVNC*. O *RealVNC* possui mais funcionalidades, porém é uma versão mais comercial, existindo apenas para *Windows*. O *TightVNC* é gratuito e inteiramente aberto, possuindo versões para *Windows* e *Linux*. Apesar de não possuir tantas funcionalidades como o *RealVNC*, o *TightVNC* possui um sistema de compressão que garante tempos de atualização de tela menores. Isso resulta em um aumento de processamento no cliente.

2.3.3 Funcionamento

O VNC é um típico aplicativo cliente-servidor e, em função disso, é dividido em dois módulos: *VNC Server* e *VNC Viewer*. O primeiro módulo deve estar em execução no computador que se pretende controlar remotamente. Já o segundo módulo é o programa que deve estar presente no computador do usuário que pretende controlar o servidor remotamente. Vale lembrar que não há a necessidade de ser utilizado o mesmo sistema operacional nesses computadores, cada um pode possuir um diferente.

Em uma conexão típica entre servidor e cliente, têm-se as seguintes etapas (SOUSA, 2004):

- Inicia-se o *VNC Server* no computador que será controlado, e uma senha de acesso precisa ser definida;
- Inicia-se o *VNC Viewer* em qualquer outro computador da internet;
- Digita-se o nome ou o IP do computador servidor na caixa de conexão do *VNC Viewer*;
- Digita-se a senha de conexão com o computador servidor no programa cliente.

[14] http://www.realvnc.com/products/download.html

Se tudo der certo, uma janela contendo uma visão com 256 cores da tela do computador que está executando o *VNC Server* abrirá na máquina cliente (SOUSA, 2004). Qualquer operação de mouse ou teclado que for feita nesta janela será transmitida e espelhada no computador remoto.

O VNC possui dois modos de operação denominados de modo ativo e passivo. No primeiro, o cliente além de visualizar o ambiente remoto, pode controlar totalmente a sessão, com todos os *inouts* sendo direcionados ao servidor. Em contrapartida, no segundo modo o cliente pode apenas visualizar o ambiente do servidor.

2.3.4 Arquitetura

O VNC é baseado no protocolo RFB (*Remote FrameBuffer*) para disponibilizar a visualização de uma interface gráfica de um computador para outro. O RFB funciona no nível de *framebuffer* (memória onde são armazenados os quadros gerados pelo sistema de vídeo), capturando o *buffer* da placa de vídeo que contém um *frame* completo de dados (BAPTISTA, 2006). É por isso que é possível haver versões do servidor VNC para diversas plataformas, como: *Linux*/*Unix*, *Windows* e *MacOS* (RICHARDSON, 2000).

Para que o protocolo RFB funcione, é preciso que haja três elementos principais: um servidor RFB, um cliente RFB e o protocolo em si, como ilustrado na Figura 4. Este último consiste basicamente na primitiva "coloque o *pixel* com estas características na posição X, Y" (RICHARDSON, 2000).

Figura 4 - Funcionamento do protocolo RFB.

O cliente RFB é responsável por enviar todos os eventos de *mouse* e telado para o servidor

RFB. Já este último, captura o *framebuffer* da interface gráfica da máquina em que está presente e o envia para o cliente, além de receber todos os eventos de teclado e *mouse* e repassá-lo ao sistema operacional, que por sua vez, os processa e atualiza a tela (RICHARDSON, 2000).

Todas as versões do VNC (*RealVNC*, *TightVNC*, *UltraVNC*, *TurboVNCiv*, etc.) devem, como implementação do protocolo RFB, prover em linhas gerais todas as características descritas nos parágrafos anteriores. O que difere uma versão de outra é que cada uma implementa, à sua maneira, as melhorias que seus desenvolvedores consideram importantes sobre o RFB. Existem versões que são próprias para redes com conexões de baixa banda, e para isso, aplicam o algoritmo de compressão a cada *frame*, além de reduzirem a quantidade de *frames* por segundo enviada ao cliente. Em contraste com essas versões, existem as que se preocupam com a fidelidade na exibição da interface gráfica. Estas, ao contrário das primeiras, transmitem uma alta taxa de *frames* por segundo. Ainda existem versões que se preocupam com a escalabilidade, sendo próprias para trabalhar com um grande número de clientes conectados simultaneamente, e para isso utilizam técnicas de transmissão como *multicast* (BAPTISTA, 2008).

2.3.5 Características do VNC

Os aspectos positivos do VNC englobam o fato de ele ser um protocolo aberto, possuir um cliente pequeno (armazenamento) que pode ser executado em diversos tipos de dispositivo computacional, incluindo dispositivos móveis. Além disso, o VNC possui implementações tanto do cliente quanto do servidor para a maioria dos sistemas operacionais, explicando o fato de ele ser tão disseminado.

O VNC apresenta como pontos negativos o fato de usar de forma ineficiente os recursos de rede e de ser um protocolo limitado, não permitindo extensões de funcionalidades sem a utilização de outros protocolos.

Outra desvantagem do VNC é que o protocolo RFB não fornece uma segurança adequada. Mesmo as senhas de autenticação sendo enviadas encriptadas, sua criptografia é facilmente

quebrada por métodos de força bruta. Além disso, é possível que uma pessoa mal intencionada capture os pacotes de *framebuffer* que o servidor envia ao cliente e, com isso, possa remontar o ambiente remoto em sua máquina. Desta forma, um atacante age como um cliente não autorizado (REALVNC, 2009).

Como uma forma de adicionar um nível maior de segurança na transmissão de dados, o VNC pode ser utilizado sobre um túnel SSH ou uma conexão SSL[15], entregando a tarefa de criptografia da conexão a um *software* mais especializado, como visto na Figura 5 (BAPTISTA, 2008).

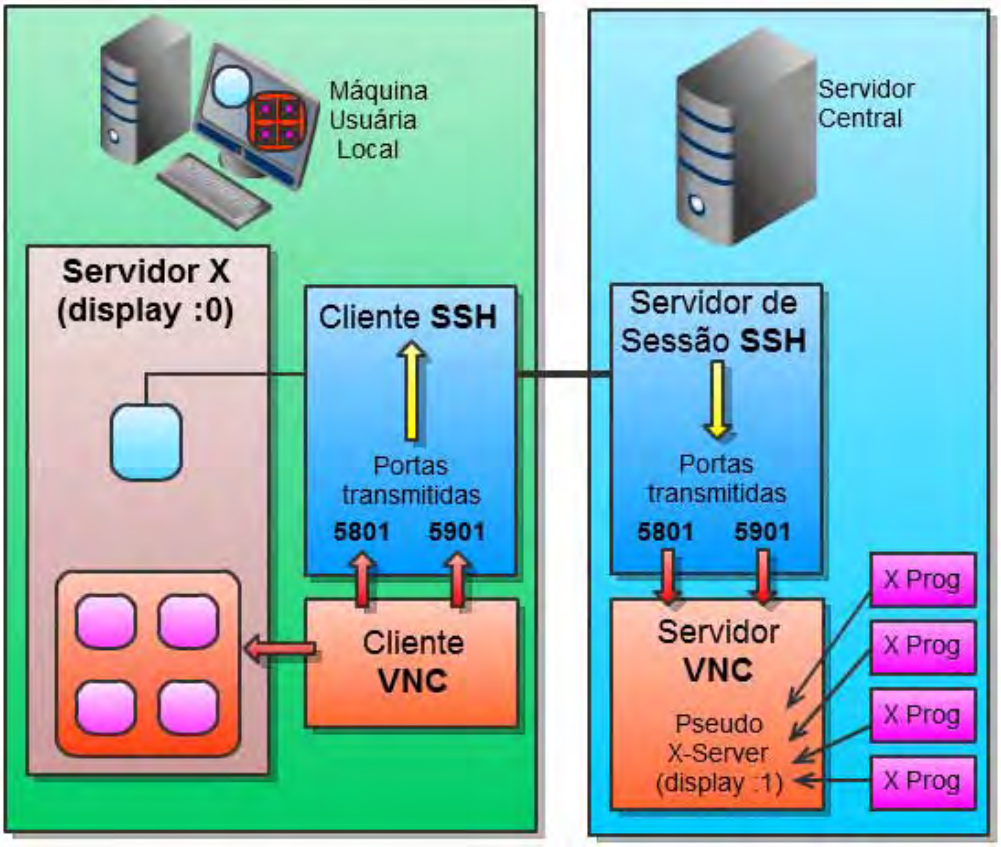

Figura 5 - Utilização do VNC sobre um túnel SSH. Adaptado de (BAPTISTA, 2008).

[15] Método de segurança de transações efetuadas via Internet. Utiliza um método de criptografia por chave pública, encriptando toda a transmissão entre o cliente e o servidor. Muito utilizado em páginas de comércio eletrônico.

2.4 NX

2.4.1 Descrição

O NX pretende ser o sucessor do VNC, permitindo acesso a *desktops* remotos com mais facilidade, mais recursos e com um sistema mais inteligente de compressão de dados que o segundo.

2.4.2 Histórico

A exemplo do *VMware*[16], o *NXServer* é um produto comercial desenvolvido pela empresa italiana NoMachine. No início, o servidor NX era pago e o cliente era distribuído gratuitamente pelo site[17] da NoMachine. Mas utilizando da mesma idéia do *VMware*, a NoMachine disponibilizou uma versão gratuita do servidor, pensando em aumentar sua participação no mercado, ganhando assim mais espaço para vender suas soluções corporativas. No entanto, tal versão gratuita do servidor NX possui uma limitação de apenas dois usuários conectados simultaneamente ao mesmo servidor.

Apesar de o NX ser um produto comercial, todas as bibliotecas utilizadas no seu desenvolvimento são *open source*. Isso permitiu o desenvolvimento do *FreeNX Server*, uma solução gratuita e aberta para o servidor NX que combina tais bibliotecas com um conjunto de *scripts*. De forma aparentemente contraditória, a NoMachine apóia o desenvolvimento do *FreeNX* e, inclusive, contribui com o projeto. Apesar de o *FreeNX Server* ter funcionalidade similar ao *NXServer*, o segundo possui uma interface melhor acabada e menos *bugs* em seu funcionamento, além de ser mais fácil de instalar e contar com suporte oficial da NoMachine. Com isso, o público corporativo prefere pagar pelo servidor comercial, enquanto o *FreeNX* possui como adeptos uma massa de usuários e colaboradores, que ajudam no desenvolvimento de suas bibliotecas base.

[16] https://www.vmware.com/
[17] http://www.nomachine.com/download.php

2.4.3 Funcionamento

Basicamente, o NX permite que clientes acessem remotamente servidores *Linux* e *Solaris*. Tais clientes podem estar fazendo o acesso através de plataformas *Windows*, *Linux* e *Solaris*.

Em comum com o VNC, o NX por padrão também exibe uma janela contendo o *desktop* do servidor, mas também apresenta a possibilidade de executar um comando específico no mesmo, o que permite abrir apenas algum programa específico. O tamanho da janela é ajustável e cada sessão é independente, o que permite que muitos clientes se conectem ao mesmo servidor usando usuários diferentes (MORIMOTO, 2006).

O diferencial da tecnologia NX com relação aos demais ambientes de trabalho remoto é que ele apresenta recursos até então não providos pelas demais tecnologias. Dentre esses recursos tem-se a transmissão de som e arquivos, o uso de aplicativos multimídia e a execução de jogos e *software* de interatividade, tais como serviços de troca de mensagens. O NX também permite aceso a navegadores (internet) e até mesmo a possibilidade de abertura de seções remotas dentro de seções remotas (OLIVEIRA et al., 2006).

Enquanto a maioria dos ambientes de trabalho remoto tira *screenshots* da tela, comprimem as imagens e transmite aos clientes, o servidor NX inicializa seções remotas do X, onde são transmitidas as instruções e os *pixmaps* necessários para montar a tela que será exibida no cliente. Esses dados são compactados através de um eficiente algoritmo próprio e encriptados pelo SSH, tornando o NX mais seguro e mais rápido que todos os ambientes de trabalho remoto, tanto utilizando a internet quanto em redes locais, onde a banda disponível é maior.

2.4.4 Arquitetura

A arquitetura do NX é distribuída, composta de uma gama de tecnologias *open source* com intuito de se obter a maior abrangência possível através de seus recursos já citados.

A tecnologia escolhida para ser o alicerce desta arquitetura distribuída do NX é o conhecido e largamente usado *X Window System*, já apresentado por este trabalho. Por isso, o servidor

NX só pode estar presente em sistemas baseados em *Unix*. Em função disso tem-se a presença dos protocolos de comunicação do X no NX.

Uma das grandes inovações apresentadas pela NoMachine no NX é o seu exclusivo protocolo de compressão (baseado no já tão conhecido *Zlib*[18]) que trabalha em conjunto com agentes integrados (REGIS, 2007), possibilitando assim acessar seções completas do *desktop* remoto, inclusive no formato *full-screen*, usando limitadíssimas larguras de banda, como conexões através de modem.

Como citado anteriormente, o servidor envia instruções, texto e componentes gráficos (ícones, imagens, gráficos, etc.) que são usados para montar a tela no cliente. O algoritmo de compressão criado pela NoMachine é otimizado para este cenário, tratando de forma diferente as imagens, os textos e as instruções e aplicando seletivamente o tipo de compressão mais adequado para cada tipo. Todas as imagens, ícones e elementos gráficos são compactados em *JPG* ou *PNG*[19], os textos são compactados usando o *Zlib* e assim por diante (MORIMOTO, 2006).

Outro aspecto interessante do protocolo do NX é que em imagens ou textos que sofreram poucas modificações em relação aos anteriormente recebidos, é utilizado um sistema de transferência diferencial, transmitindo apenas as partes diferentes, reduzindo significativamente a taxa de transmissão (MORIMOTO, 2006).

Outro aspecto que possibilita melhor desempenho do NX é que ele usa a idéia de prioridades da seguinte forma: instruções utilizadas para montar a tela no cliente são transmitidas com prioridade maior do que imagens. Isso faz com que a interface, no cliente, continue respondendo enquanto imagens são enviadas (MORIMOTO, 2006).

O mecanismo de compressão do NX opera em 3 níveis do protocolo X (OLIVEIRA et al., 2006):

- Comprime o tráfego da rede em vários sentidos, como por exemplo, através do uso de avançados métodos de *caching*, redução de perda e compressão de imagens;
- Reduz o chamado *network round-trip* (uma das desvantagens do protocolo X) a

[18] Biblioteca multiplataforma de compressão de dados.
[19] Formato de dados utilizado para imagens que permite compressão de imagens sem perda de qualidade.

praticamente zero, maximizando o *throughput*;

- Adaptação da largura de banda em tempo real, de acordo com as condições da rede.

Com isso, as taxas de compressão em relação ao protocolo do X varia entre 10:1 e 100:1

A Figura 6 exibe a tela com resolução 640x480 de um cliente acessando um servidor dotado do sistema operacional Kurumin[20] com o *FreeNX* via ADSL (256k). Pode-se notar que alguns pontos da imagem ficam sem definição devido à compressão via JPG, mas no geral a imagem está nítida e os tempos de resposta são menores se comparados ao VNC na mesma conexão.

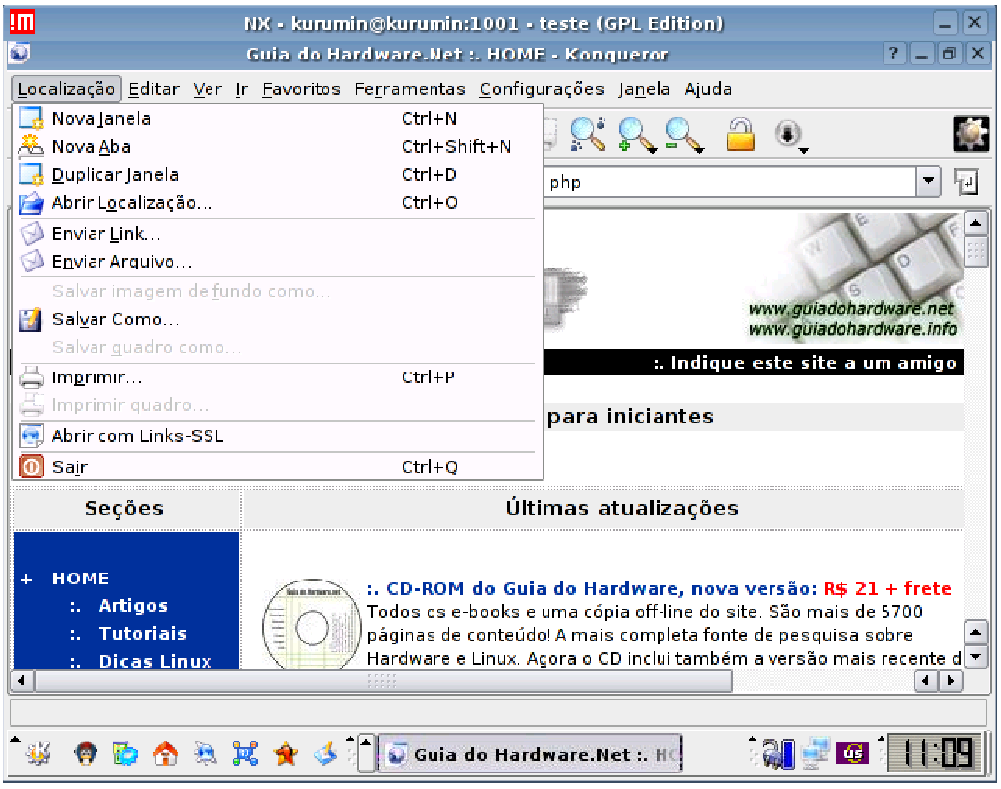

Figura 6 - Acessando um servidor com *FreeNX*. Fonte: (MORIMOTO, 2006).

Para diminuir o tráfego de dados e para contornar o problema dos "*roundtrips*" imposto pelo X, o NX utiliza dois tipos de *proxys*. O primeiro tipo de *proxy* é conhecido como "nxproxy". Há um “nxproxy” tanto no servidor (*Remote NX Proxy*) quanto no cliente (*Local NX Proxy*) (REGIS, 2007). Ele tem por finalidade criar um *cache*, tanto no servidor quanto no cliente, responsável por armazenar dados já transferidos, reutilizando-os sempre que possível. Um papel de parede, por exemplo, é transmitido apenas uma vez, já que depois de recebido, o

[20] http://www.gdhpress.com.br/kurumin/

cliente pode utilizá-lo indefinidamente. Os *caches* devem ser sincronizados entre servidor e cliente, ou seja, ambos precisam possuir o mesmo conjunto de dados. Este tipo de *proxy* ajuda muito a reduzir o trafego da rede, evitando retransmissões de dados presentes nos *caches*. Melhor ainda, tal *cache* pode ser utilizado em conexões futuras entre mesmos clientes e servidores. Com isso, a partir da segunda conexão, o volume de dados transmitidos cai brutalmente (MORIMOTO, 2006).

O segundo tipo de *proxy*, nomeado de "*nxagent*", fica no servidor NX. Sua função é receber as ações, como movimento do mouse e teclas digitadas, do cliente e transmiti-las para os aplicativos em execução no próprio servidor. A partir daí, o próprio servidor NX monta as telas e transmite os pacotes de resposta localmente, ou seja, no próprio servidor e por fim transmite as atualizações da tela para o cliente (MORIMOTO, 2006).

Desta forma o NX consegue eliminar quase que inteiramente os pacotes de resposta (“*roundtrips*”), pois eles são transmitidos localmente, sendo enviado ao cliente apenas os dados referentes à montagem da tela (REGIS, 2007).

A Figura 7 ilustra um cenário de um cliente conectado ao servidor, onde é possível uma melhor compreensão do sistema de *proxys*.

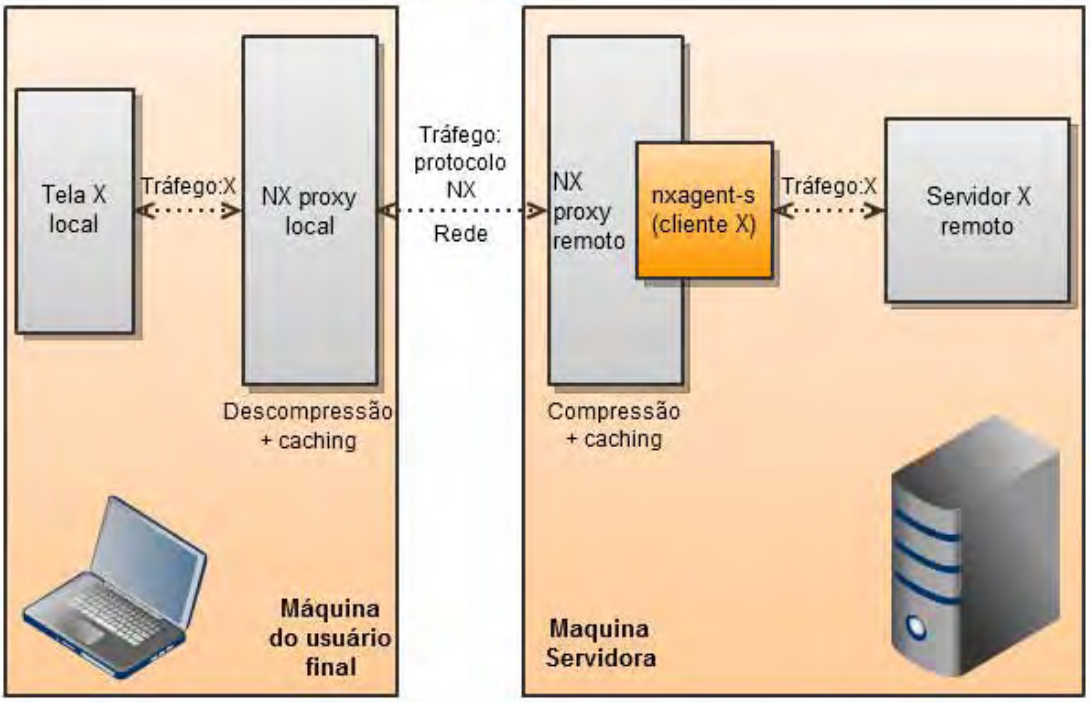

Figura 7 – Sistema de *proxys* do NX. Adaptado de (REGIS, 2007).

2.4.5 Características do NX

Atualmente, as soluções computacionais baseadas em cliente-servidor através de uma rede são adaptações para sistemas operacionais que não foram essencialmente projetados para trabalhar em rede, ou seja, tais soluções são uma forma de contornar a deficiência nativa da maioria dos sistemas operacionais. Já no caso do NX, toda sua arquitetura foi desenvolvida baseada na computação em rede e aproveitando a eficiente arquitetura dos sistemas *Unix*, reconhecidamente mais estáveis e com melhor desempenho. Com ele, é possível obter-se os benefícios da computação em rede, como criptografia, mobilidade, compressão e múltiplas plataformas de acesso (OLIVEIRA et al., 2006).

Uma limitação do NX é que ele possui servidor apenas para sistemas operacionais baseados em *Unix*, limitando um pouco seu uso. Outro aspecto negativo relacionado ao NX é que, devido ao fato de inicialmente ser pago, existe pouca documentação a seu respeito, dificultando a compreensão de sua arquitetura e funcionamento.

2.5 *NeatX*

2.5.1 Descrição

Neatx é uma distribuição *Open Source* do servidor NX, similar ao NX comercial da NoMachine (GOOGLE, 2010). A Google alega que os produtos atuais do *NX Server* são pagos, e o correspondente livre *FreeNX* é de difícil manutenção devido ao fato de ser construído sobre muitas linguagens, por isso a necessidade de se ter uma solução *Open Source* homogênea e de fácil configuração. O *Neatx* é escrito em Python, com alguns scripts Bash e, por questões de desempenho, um programa escrito em C. Por ser recente, ainda não se tem grandes acervos de informações sobre o projeto.

2.5.2 Histórico

O projeto *NeatX* foi desenvolvido pela Google para um projeto interno. Tal projeto já foi concluído, e os fontes foram liberados para que a comunidade possa usar, desenvolver novidades e contribuir com o projeto (GOOGLE, 2010). Um grupo de funcionários da Google, está continuamente lançando atualizações para o *NeatX*.

2.5.3 Funcionamento

O *NeatX* permitirá o básico para um Servidor de Terminais: criar sessões, suspendê-las, retomá-las e desligá-las. Há suporte ao GNOME, KDE e qualquer aplicação que rode no terminal. Ele também suporta janelas flutuantes e sessões virtuais, tela cheia, resolução personalizada, e preferências de teclado.

O projeto implementará funcionalidades não presentes no *FreeNX* e nem todas as ferramentas do *FreeNX* estarão presentes no *NeatX*, pelo menos nessa fase inicial.

2.6 Quadro Comparativo Entre as Tecnologias

A Tabela 1 apresenta uma comparação entre as tecnologias apresentadas por este trabalho, com exceção do *NeatX*, já que este consiste em um projeto muito recente que ainda não apresenta uma documentação completa.

Tabela 1 - Comparativo entre as Tecnologias.

	GRA	PRO	OSC	UTL	ROB	MUL	FLEX	LOC	INT	SSH	COM
X	√		√		√			√		√	
RealVCN	√			√				√	√	√	
TightVNC	√		√	√			√	√	√	√	
NXServer		√				√		√	√	√	√
FreeNX	√		√			√		√	√	√	√

Legenda:

GRA - Gratuito
PRO - Proprietário
OSC – *Open Source*
UTL - Simplicidade na Utilização
ROB - Robustez
MUL - Suporte a Aplicações Multimídias
FLE - Flexibilidade
LOC - Apropriado para Rede Local
INT - Apropriado para Internet
SSH – Possibilidade de Encriptar o Tráfego com o SSH
COM - Sistema Otimizado de Compressão de Dados

Através da Tabela 1 é possível observar que as ferramentas baseadas na tecnologia NX (*NXServer* e *FreeNX Server*) são as mais completas e as únicas que oferecem funcionalidade especificamente projetada para redução na taxa de transmissão de dados sendo, desta forma, mais indicadas para redes com baixas taxas de transmissão, como as redes sem fio. Como única limitação com relação às ferramentas VNC, a tecnologia NX oferece acesso remoto apenas a sistemas baseados em *Unix*.

2.7 Considerações Finais

Este trabalho apresentou uma revisão bibliográfica sobre ambientes de trabalho remoto. Os

tópicos principais aqui abordados são: servidores de terminais, *X Window System*, *Virtual Network Computer*, tecnologia NX e *NeatX*. Os ambientes mais utilizados atualmente são o *X Window System*, que continua sendo muito utilizado para soluções *Thin Client*, e o VNC, que é muito utilizado para assistência remota. O NX ainda vem ganhando espaço no mercado e o *NeatX* é muito recente e ainda precisa se firmar como uma solução para acesso a *desktops* remotos, porém a tendência é de que esses dois últimos venham a possuir a maior parcela do mercado no futuro.

A configuração de um parque informático utilizando um sistema de servidores de terminais permite a reutilização de máquinas consideradas obsoletas como terminais comuns, havendo a necessidade apenas de se investir em sistemas mais potentes para os servidores, que são os responsáveis por todo o armazenamento das aplicações e por todo o processamento. Tal configuração é mais econômica, já que ao invés de se investir em melhoria de *hardware* para diversos clientes, há a necessidade de apenas melhorar o servidor.

Além do aspecto econômico, o uso de servidores de terminais em conjunto com clientes proporciona mais facilidade de gerenciamento de TI, já que instalações e manutenções vão passar a serem realizadas apenas uma vez, no servidor, em oposição às dezenas de modificações individuais nos clientes, que são típicas em diversos cenários empresariais e acadêmicos.

Outro benefício introduzido pelo uso de servidores de terminais é o aumento na segurança de dados. Informação descentralizada permite a possibilidade de fraudes, perda de integridade das bases de dados (como por exemplo, dois clientes possuindo dados diferentes relacionados às mesmas informações) e perda de dados por problemas de *hardware* ou por erros de usuário.

Em contrapartida, se o servidor de aplicações apresentar uma falha, todos os terminais ficam inoperáveis. Com isso, dependendo das exigências do ambiente em que se quer configurar o servidor, há a necessidade de se garantir um determinado nível de disponibilidade que respeite as exigências impostas.

3. VIRTUALIZAÇÃO

3.1 Introdução

Virtualização é um conceito primeiramente desenvolvido na década de 60 para compartilhar *hardware* de grandes *mainframes*. Nessa época, era comum que cada computador (*mainframe*), mesmo de um único fabricante, tivesse seu próprio sistema operacional, e isso se tornou uma das principais razões para o aparecimento das máquinas virtuais: permitir que *software* legado executasse nos caros *mainframes*. Essa abordagem foi usada com sucesso pela IBM que, na linha de *mainframes* 370 e seus sucessores, oferecia uma máquina virtual, portada para várias de suas plataformas, sobre a qual as aplicações executavam (CARISSIMI, 2008). Dessa forma era possível executar, ou migrar, uma aplicação de uma plataforma para outra desde que houvesse uma versão de máquina virtual para a plataforma alvo.

À medida que os computadores começaram a se tornar mais comuns, a quantidade de sistemas operacionais convergiu para algumas poucas famílias (Unix, Macintosh e Microsoft), cada uma com um público-alvo e um conjunto de aplicativos (CARISSIMI, 2008). Com isso, a heterogeneidade dos sistemas deixou de ser um enorme problema e a virtualização passou a perder utilidade neste cenário. No entanto, o aumento do poder computacional dos atuais processadores, a disseminação de sistemas distribuídos e a ubiquidade das redes de computadores causaram, por várias razões, o ressurgimento da virtualização.

Com o passar dos anos, a capacidade de processamento dos computadores vêem aumentado rapidamente, porém toda a capacidade adquirida não tem sido totalmente aproveitada. Há situações em que na maioria do tempo que poderia ser usado para o processamento de aplicações, o servidor fica ocioso. Uma solução é a utilização de máquinas virtuais, que estão cada vez mais sendo adotadas por empresas em geral e por empresas de pequeno e médio porte (ANDRADE, 2006). A vantagem de se utilizar virtualização é que há

um maior aproveitamento dos computadores, reduzindo desta forma os custos das empresas.

A virtualização permite que em uma mesma máquina sejam executadas simultaneamente dois ou mais ambientes distintos e isolados. Do ponto de vista prático, a virtualização é uma tecnologia que permite que um único computador seja utilizado como se fosse vários computadores, chamados de máquinas virtuais, todas podendo ser executadas simultaneamente e uma não interferindo no funcionamento da outra (ALKMIM, 2009). Cada máquina virtual possui seus próprios recursos, que são provenientes do computador físico, e pode executar seu próprio sistema operacional e suas próprias aplicações. Desta forma, os recursos ociosos do computador físico passam a ser melhor aproveitados.

Contudo, atualmente o interesse na virtualização não se atém somente ao fato de permitir o uso de um mesmo sistema por vários usuários concomitantemente, mas os principais interesses são a segurança, confiabilidade e disponibilidade, custo, adaptabilidade, balanceamento de carga e suporte a aplicações legadas (MATTOS, 2008).

Nos últimos anos, dada a importância e a gama de aplicações em que a virtualização pode ser empregada, houve um enorme investimento nesta tecnologia por parte de fabricantes de processadores e no desenvolvimento de produtos de *software* (CARISSIMI, 2008). Os processadores mais recentes da Intel e da AMD contam no seu projeto com mecanismos e soluções de *hardware* especialmente destinadas a dar suporte a virtualização, o que vem melhorando cada vez mais o desempenho de sistemas virtualizados.

3.2 Máquinas Virtuais

Uma máquina virtual (MV) nada mais é do que uma abstração, em *software*, de uma máquina física real. Isso permite a criação de máquinas virtuais com os seus próprios dispositivos virtuais através do particionamento de uma única plataforma de *hardware*, onde tais dispositivos são exclusivos de cada máquina virtual.

Dentre os principais conceitos envolvidos no estudo de máquinas virtuais tem-se o de um monitor (também chamado de *hypervisor*) VMM (*Virtual Machine Monitor*). O monitor é uma camada de *software* inserida entre o sistema virtual, SO convidado (SOC), e o *hardware* da máquina física. Essa camada faz uma interface entre os possíveis sistemas virtuais (SOCs) e o *hardware* que é compartilhado por eles. O VMM é responsável por gerenciar todas as estruturas de *hardware*, como MMU[21], dispositivos de Entrada/Saída, controladores DMA[22], criando um ambiente completo, onde os sistemas virtuais executam.

Segundo (KING et al., 2003), VMMs podem ser classificados de diversas maneiras. Uma delas refere-se à proximidade da interface que eles provêem do *hardware* subjacente. VMMs como o *VMware ESX Server* (VMWARE, 2002) e emuladores como o *Bochs* apresentam uma abstração de *hardware* idêntica ao *hardware* subjacente (ANDRADE, 2006). Para chegar a um nível de abstração tão alto, alguns monitores inserem *device drivers* no SOH (Sistema Operacional Hospedeiro) que serão usados pelo SOC através do VMM. Já outros VMMs apresentam uma interface diferente do *hardware* subjacente, como por exemplo, a máquina virtual *Java* tem uma arquitetura totalmente independente do *hardware* sobre o qual executa.

Outra classificação de VMMs se refere à plataforma sobre a qual eles executam (GOLDBERG, 1973). Segundo essa classificação, as máquinas virtuais podem ser classificadas em dois tipos (KING et al., 2003):

- Arquitetura de Máquina Virtual Não Hospedada: O VMM é desenvolvido diretamente sobre o *hardware* físico do computador. Os VMMs *Xen* (XEN SOURCE, 2010) e *VMware ESX Server* são exemplos desse tipo de máquinas virtuais;
- Arquitetura de Máquina Virtual Hospedada: O VMM é desenvolvido completamente sobre o sistema operacional hospedeiro. Exemplos desse tipo de máquinas virtuais são o *VMware Server* (VMWARE, 2011a), o *VirtualPC* e o *VirtualBox*.

As Figuras 8 e 9 apresentam esquemas das arquiteturas citadas anteriormente.

[21] Unidade de Gerenciamento de Memória (MMU, em inglês) é um dispositivo de hardware que transforma endereços virtuais em endereços físicos.
[22] O termo DMA é um acrônimo para a expressão em inglês *Direct memory access*. O DMA permite que certos dispositivos de hardware num computador acedam a memória do sistema para leitura e escrita independentemente da CPU.

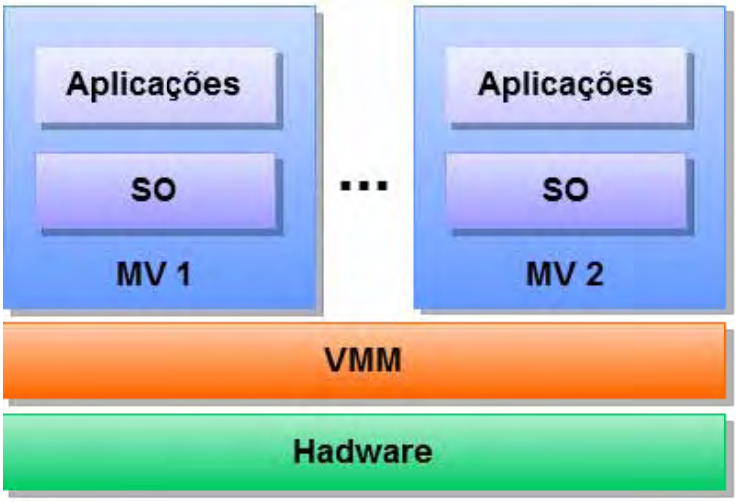

Figura 8 - Arquitetura de Máquina Virtual Não Hospedada.

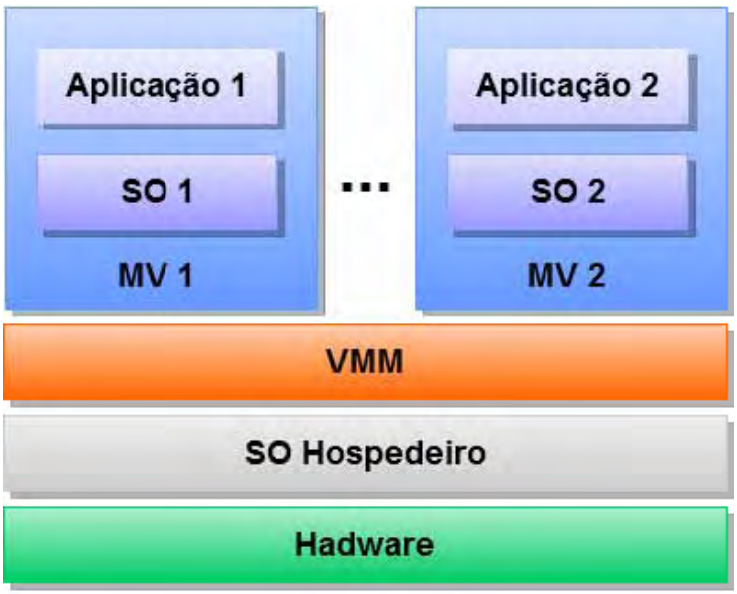

Figura 9 - Arquitetura de Máquina Virtual Hospedada.

Algumas alterações podem ser inseridas nas arquiteturas descritas anteriormente com a finalidade de aumentar o desempenho das máquinas virtuais. Essas modificações geram arquiteturas híbridas. Exemplos de tais arquiteturas podem ser encontrados em emuladores, apresentados na seção a seguir.

3.3 Emuladores

Um emulador é um sistema que simula toda uma arquitetura computacional. Nele, todas as instruções do sistema operacional emulado são traduzidas e executadas no SO hospedeiro (SOH). Geralmente, emuladores são usados para testar o funcionamento de programas escritos para arquiteturas diferentes daquelas em que o emulador executa. A diferença básica entre emuladores e máquinas virtuais, está no nível de abstração. Enquanto, em um emulador todas as instruções são convencionalmente traduzidas e interpretadas, em MVs as instruções são executadas no modo mais nativo possível.

Emuladores podem ser classificados de diversas formas. Segundo (LAUREANO, 2006) podem ser classificados, quanto à sua natureza de uso, da seguinte forma:

- Emuladores de processador;
- Emuladores de sistemas operacionais;
- Emuladores de uma plataforma de *hardware* específica;
- Emuladores de videogames (consoles).

Como exemplo de emuladores de processadores tem-se o *Bochs*[23], que emula processadores da linha x86. Seu desempenho não é muito bom, pois as instruções são interpretadas e executadas uma por uma (BJERKE, 2005). Sistemas como o *Bochs* são mais usados para simulação, tendo mais utilidade em projetos de *software* baixo nível (como sistemas operacionais).

Uma alternativa à tradução "instrução-por-instrução" é a tradução de blocos de instruções para posterior execução nativa (ANDRADE, 2006). Com essa abordagem faz-se necessária a re-tradução para blocos de código já traduzidos. Exemplos de emuladores que se utilizam dessa técnica são o *Microsoft Virtual PC* (MICROSOFT, 2011) e QEMU (BELLARD, 2005). Há também os emuladores de consoles (videogames), como o *ePSXe*[24], o *SNES9x*[25] e o *ZSNES*[26]. Outro exemplo ainda é o *WINE*[27], emulador de bibliotecas de usuário.

[23] http://bochs.sourceforge.net
[24] http://www.epsxe.com
[25] http://www.snes9x.com
[26] http://www.zsnes.com
[27] http://www.winehq.org

3.4 Técnicas de Virtualização

A técnica de virtualização tem grande importância para projetos de sistemas computacionais, utilizada com o objetivo principal de desacoplar os recursos físicos de um *hardware* da sua representação lógica (BARHAM et al., 2003). Existem várias técnicas usadas na virtualização. As principais são virtualização completa, paravirtualização e recompilação dinâmica. Tais técnicas são detalhadas nos subtópicos seguintes.

3.4.1 Virtualização completa

A virtualização completa (Figura 10) tem por objetivo fornecer ao sistema operacional convidado uma réplica da infra-estrutura de *hardware* físico presente no computador. Dessa forma, o sistema operacional visitante é executado sem modificações sobre o monitor de máquina virtual (VMM), o que traz alguns inconvenientes.

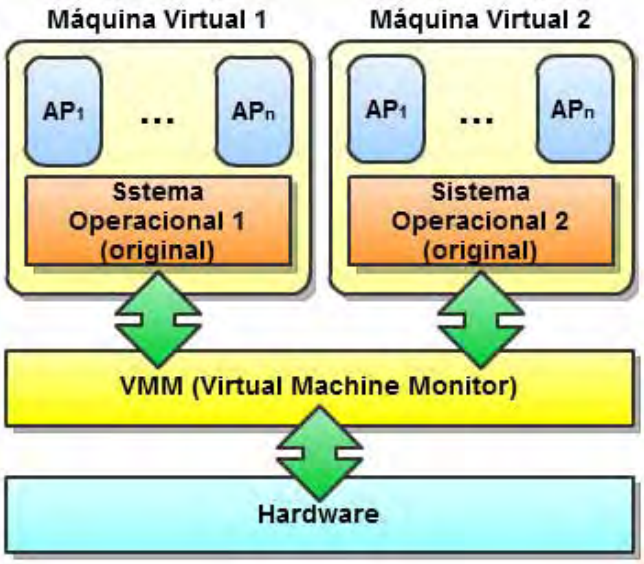

Figura 10 - Virtualização Completa. Adaptado de (CARISSIMI, 2008).

A primeira desvantagem é que o número de dispositivos a serem suportados pelo VMM é extremamente elevado. Para contornar esse impasse, as implementações da virtualização completa usam dispositivos genéricos, que apresentam um bom desempenho para a maioria

dos dispositivos disponíveis, mas não garantem o uso das capacidades máximas suportadas por estes (MATTOS, 2008).

Outro inconveniente da virtualização completa vem pelo fato de o sistema operacional visitante não ter conhecimento de que está sendo executado sobre o VMM. Desta forma as instruções executadas pelo sistema operacional visitante devem ser anteriormente validadas pelo VMM para que depois sejam executadas diretamente no *hardware*, ou executadas pelo VMM e simulada a execução para o sistema visitante (MATTOS, 2008).

Por fim, o último inconveniente da virtualização completa é ter que contornar alguns problemas gerados pela implementação dos sistemas operacionais, já que esses foram desenvolvidos para serem executados nativamente sobre máquinas físicas, não havendo disputas por recursos entre diferentes sistemas operacionais. Um exemplo deste último inconveniente é uso de paginação na memória virtual, pois há a disputa de recursos entre diversas instâncias de sistemas operacionais, o que acarreta em uma queda do desempenho (MATTOS, 2008).

3.4.2 Paravirtualização

A paravirtualização (Figura 11) é uma abordagem alternativa que surge como forma de contornar as desvantagens da virtualização completa (CARISSIMI, 2008). Este método consiste em apresentar ao SO convidado uma arquitetura virtual que é similar, mas não idêntica à arquitetura física real (XEN SOURCE, 2010). Essa solução aumenta o desempenho das máquinas virtuais que a utilizam (BARHAM et al., 2003). Em contrapartida, o sistema convidado, que executa na arquitetura x86, precisa ser modificado. Tais alterações já são implementadas nas versões recentes do *Linux* e *Windows Server*.

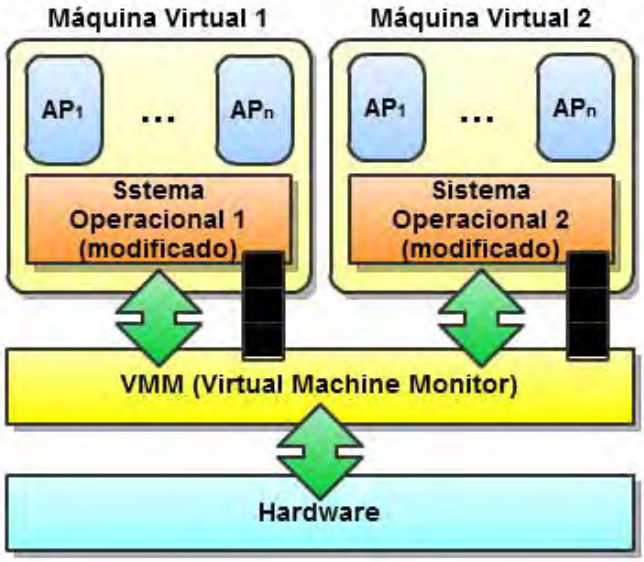

Figura 11 - Paravirtualização. Adaptado de (CARISSIMI, 2008).

A virtualização completa permite que um sistema convidado execute em um VMM sem que o primeiro precise ser alterado, ao passo que com a paravirtualização o sistema convidado deve ser modificado para enxergar o VMM. Por outro lado, a paravirtualização explora de maneira apropriada os recursos disponíveis pelo *hardware* real da máquina e apresenta um desempenho superior à virtualização completa. Entretanto, com o atual suporte de *hardware* à virtualização presente nos processadores Intel e AMD, a virtualização completa e a paravirtualização têm apresentado desempenhos semelhantes (CARISSIMI, 2008).

3.4.3 Recompilação dinâmica

Uma técnica frequentemente utilizada, inclusive utilizada na virtualização completa, é a tradução dinâmica (*dynamic translation*) ou recompilação dinâmica (*dynamic recompilation*) (LAUREANO, 2006). Nesta técnica o VMM analisa, reorganiza e traduz sequências de instruções geradas pelo sistema convidado em novas sequências de instruções (TEIXEIRA, 2010). Uma aplicação da técnica é vista em compiladores JIT (*just-in-time*), que traduzem de uma linguagem *bytecode* para código nativo da CPU onde o compilador executa.

A tradução dinâmica poderá ser (TEIXEIRA, 2010):

- Binária. A entrada do tradutor é código binário x86, e não código fonte;

- Dinâmico: A tradução acontece em tempo de execução, intercalando com a execução do código gerado;
- *On demand*: O código é traduzido somente quando está para ser executado. Esta execução passo a passo elimina a necessidade de estruturas para armazenar código e dados;
- *Subsetting*: A entrada do tradutor é completada com um conjunto de instruções x86, incluindo todas as instruções privilegiadas[28]. A saída do tradutor é composta por um conjunto seguro de instruções, normalmente instruções no modo usuário;
- Adaptativa: O código traduzido é ajustado em resposta a mudanças de comportamento do convidado para melhorar a eficiência de modo geral.

A recompilação (ou tradução) é feita basicamente seguindo os seguintes passos (ANDRADE, 2006):

- Passo 1: O código binário é escaneado para que seja identificado uma sequência de bits correspondente à seção de código do programa em execução;
- Passo 2: Os bits agrupados anteriormente são divididos em instruções, juntamente com os parâmetros delas;
- Passo 3: As instruções são transformadas para uma representação mais próxima da máquina nativa;
- Passo 4: Um código em uma linguagem de alto nível é gerado a partir da representação anterior, código esse que é compilado e reescrito na linguagem nativa. Assim, tem-se uma sequência de bits executável no *hardware* nativo.

Emuladores como o QEMU utilizam essa técnica para aumentar seu desempenho (BELLARD, 2005). O *VMware Workstation* (VMWARE, 2011b) também utiliza essa técnica, recompilando apenas parte do código, já que boa parte dele pode executar nativamente (a arquitetura de *hardware* subjacente é a mesma da máquina virtual). No *VMware Workstation* apenas instruções que não podem ser executadas diretamente são recompiladas (LAUREANO, 2006).

[28] Instruções acessíveis somente por meio de códigos executando em nível privilegiado (código de núcleo).

Outra aplicação típica que se utiliza de recompilação dinâmica é a máquina virtual *Java®*, que recebe o nome de *JIT Compiler.* As instruções geradas para a máquina virtual e armazenadas nos *bytecodes* das classes *Java* são traduzidas e executadas no *hardware* subjacente.

3.5 Virtualização Assistida por *Hardware*

Conforme visto anteriormente, a arquitetura x86 não foi projetada tendo em vista a virtualização. Com isso, fornecedores de *hardware* (como Intel e AMD) estão aderindo à idéia da virtualização e propondo novas arquiteturas que facilitem a virtualização.

Tradicionalmente, os processadores implementam 4 níveis (ou *rings*, na terminologia usada para execução de processos) (Figura 12) de proteção (JONES, 2006). Quanto maior o nível, menos privilégios para a execução de instruções são concedidos. O nível 0 (*ring-0*) é geralmente usado pelo sistema operacional, níveis 1 e 2 pelos processos do sistema operacional e o nível 3 pelas aplicações (ANDRADE, 2006).

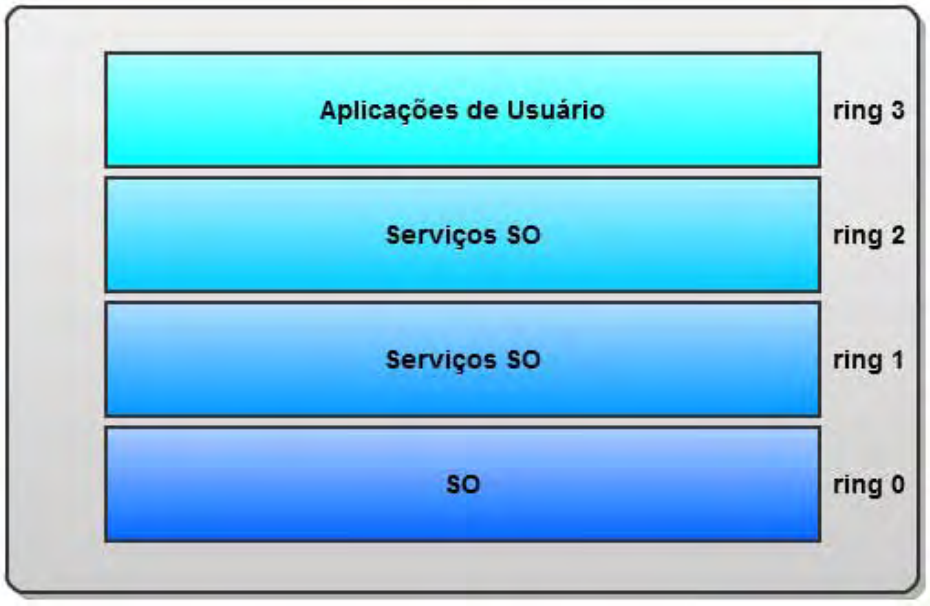

Figura 12 - Anéis (*rings*) de proteção da x86. Adaptado de (ANDRADE, 2006).

A primeira geração de arquiteturas que visam apoio a virtualização inclui a *Intel Virtualization Technology* (VT-x) e AMD-V. Em ambas existem instruções privilegiadas com

novo modo de execução da CPU que permitem que o *hypervisor* seja executado em um novo modo raiz abaixo no *ring* 0 (TEIXEIRA, 2010), como mostra a Figura 12.

Chamadas privilegiadas e sensíveis[29] são estruturadas para automaticamente causarem um *trap* para o *hypervisor,* evitando a necessidade de tradução binária ou paravirtualização (TEIXEIRA, 2010). O estado do convidado é armazenado na *Virtual Machine Control Structures* (no caso da VT-x) ou *Virtual Machine Control Blocks* (no caso da AMD-V). Processadores com Intel VT e AMD-V passaram a ser desenvolvidos em 2006.

3.6 Propriedades de VMMs

Em 1974, Popek e Goldberg (POPEK & GOLDBERG, 1974), introduziram três propriedades necessárias para que um sistema computacional oferecesse de forma eficiente suporte a virtualização (CARISSIMI, 2008):

- Eficiência: todas as instruções de máquina que não afetam o funcionamento do sistema devem ser executadas diretamente sobre *hardware* sem intervenção da máquina virtual. As instruções da máquina virtual que não puderem ser executadas pelo processador real devem ser interpretadas pelo *hypervisor* e traduzidas em ações equivalentes no processador real, caso contrário, devem ser executadas diretamente no *hardware*;
- Controle de recursos: uma máquina virtual deve ter controle completo sobre os recursos virtualizados sendo estritamente proibido que um programa executando sobre a máquina virtual os acesse diretamente. Além disso, a qualquer momento o *hypervisor* pode resgatar recursos alocados;
- Equivalência: um programa executando sobre uma máquina virtual deve apresentar um comportamento idêntico àquele apresentado caso a máquina virtual não existisse e o programa fosse executado diretamente sobre uma máquina física equivalente. Duas exceções são consideradas. Primeira, eventualmente, algumas instruções

[29] Instruções que podem consultar ou alterar o status do processador, ou seja, os registradores que armazenam o atual status da execução na máquina real.

podem ter seu tempo de execução aumentado. Segunda, pode haver problemas de conflito de acesso a recursos, os quais devem ser resolvidos de forma apropriada.

Além destas propriedades, outras características derivadas são associadas ao *hypervisor* (ROSENBLUM, 2004):

- Isolamento: garantir que um processo em execução em uma máquina virtual não possa interferir no funcionamento de outro processo em execução, tanto no monitor quanto em outra máquina virtual. Também é necessário que o VMM gerencie se o funcionamento de um processo numa máquina virtual está comprometendo o desempenho de outras máquinas virtuais;
- Inspeção: o *hypervisor* deve ser capaz de acessar e controlar todas as informações dos processos em execução na máquina virtual, como memória, estado da CPU, eventos, etc.;
- Gerenciabilidade: como cada VM é uma entidade das demais, deve-se ter a possibilidade de gerenciar uma máquina virtual independente de outra VM;
- Encapsulamento: o *hypervisor* pode controlar totalmente processos em execução em VMs, podendo gerar *"checkpoints"* do sistema para possíveis recuperações pós-falhas;
- *Recursividade*: deve-se ter a possibilidade de executar um *hypervisor* dentro de uma máquina virtual, produzindo um novo nível de máquinas virtuais. Neste caso, a máquina real é normalmente denominada *máquina de nível 0.*

3.7 Ferramentas de Virtualização

Aqui serão discutidas algumas das principais ferramentas de virtualização disponíveis atualmente: *VMware*, *Xen*, *Virtual PC*, QEMU, KVM[30] e *VirtualBox*.

[30] http://www.linux-kvm.org/page/Small_look_inside

3.7.1 *VMware*

É o projeto de *software* mais desenvolvido e divulgado atualmente. A *VMWare* fornece uma família de produtos para virtualização da arquitetura x86 destinadas a diferentes tipos de uso:

- *VMware Server*: é a versão gratuita dos produtos *ESX Server*. Tem por objetivo principal permitir que usuários testem o produto antes de adquiri-lo. Assim como as versões *ESX*, o *VMware Server* oferece a virtualização de processador, memória, armazenamento e infra-estrutura de rede que são configurados através de ferramenta própria não disponível na versão gratuita (CARISSIMI, 2008). Para contornar o problema de configuração, ou seja, a criação do ambiente hóspede, a *VMware* oferece uma série de imagens de ambientes predefinidas em seu site (CARISSIMI, 2008). Tais imagens são denominadas de *appliances* e possuem os serviços de rede mais comuns (web, arquivos, impressão, DNS, etc.);
- *VMware Workstation* (anteriormente *VMware GSX Server*): é versão que permite a criação de máquinas virtuais sobre o *hypervisor*. Isso significa que é possível carregar um sistema operacional qualquer nessa máquina virtual e executar as suas aplicações (CARISSIMI, 2008). A configuração das máquinas virtuais para um determinado sistema operacional é feita através de ferramenta específica que é parte integrante desse produto;
- *VMware ESX Server*: é um produto destinado a *data centers*, para virtualização de recursos de *hardware* em servidores. Ele consiste em uma camada de *software* que multiplexa os recursos de *hardware* entre as máquinas virtuais. O projeto deste produto difere em muito ao *VMware Workstation*, pois o *ESX* gerencia diretamente o *hardware*, permitindo uma melhor performance de I/O e completo controle sobre o gerenciamento de recursos. Este produto também utiliza mecanismo de tradução binária para a virtualização da arquitetura x86 a semelhança do *VMWare Workstation*, mas que também pode utilizar as extensões de virtualização da Intel e da AMD. O VMM utiliza mecanismos de gerenciamento de memória mais sofisticados, que inclusive permite o compartilhamento de paginas de memória baseadas em seu conteúdo entre diferentes máquinas virtuais (WALDSPURGER, 2002);

- *VMware Player*: é a versão gratuita do produto VMware *Workstation*. Assim como ocorria na versão paga, o objetivo é permitir que usuários testem o uso da virtualização e não é possível definir (criar) o sistema hóspede na máquina virtual a partir do zero (CARISSIMI, 2008). A *VMware* também distribui, para esta ferramenta, uma série de *appliances* (imagens de sistemas operacionais) de diferentes distribuições de *Linux* e *Windows Server 2003*;
- *VMware Fusion*: é a solução *VMware Workstation* equivalente para o sistema operacional *MacOS X*.

3.7.2 *Xen*

O projeto *Xen* nasceu no Laboratório de Computação da Universidade de Cambridge, com o objetivo de implementar um VMM de alta performance para a plataforma x86. O *Xen* consiste em um monitor de máquina virtual (*hypervisor* ou VMM), em *software* livre, licenciado nos termos da *GNU General Public Licence* (GPL), para arquiteturas x86, que permite vários sistemas operacionais hóspedes serem executados em um mesmo sistema hospedeiro (CARISSIMI, 2008).

O *Xen* é popular na virtualização de SO *Unix*, isso porque originalmente o *Xen* realizava apenas a paravirtualização, que exige a modificação do SO convidado, o que só se consegue nos SO de código aberto. A modificação é necessária somente no *kernel* do SO convidado. Essa popularidade é tão grande que algumas versões de sistemas *Linux*, como o *SUSE*, vêm com módulos do *Xen* (XEN SOURCE, 2010).

Versões recentes do sistema *Xen* passaram a apresentar o suporte de virtualização disponível nos processadores atuais, tornando possível a execução de sistemas operacionais convidados sem modificações (como o *Windows*, por exemplo), embora com um desempenho ligeiramente menor que no caso de sistemas paravirtualizados (TEIXEIRA, 2010). Conforme seus desenvolvedores, o custo e impacto das alterações nos sistemas convidados são baixos e a diminuição do custo da virtualização compensa essas alterações (a degradação média de desempenho observada em sistemas virtualizados sobre a plataforma *Xen* não excede 5%) (XEN SOURCE, 2010).

O *Xen* realiza a virtualização de uma forma diferente das demais ferramentas. Os dois principais conceitos do *Xen* são domínios e *hypervisor*. Os domínios são as máquinas virtuais do *Xen* e são de dois tipos: privilegiada (domínio 0) ou não-privilegiada (domínio U) (CARISSIMI, 2008). A finalidade do *hypervisor* é controlar os recursos de comunicação, de memória e de processamento das máquinas virtuais, e não possui *drivers* de dispositivos. Considerando as características do *hypervisor Xen*, tem-se que ele não é capaz de suportar nenhum tipo de interação com sistemas hóspedes. Em função disto, é necessário que exista um sistema inicial para ser invocado pelo *hypervisor*. Esse sistema inicial é o domínio 0. As demais máquinas virtuais apenas podem ser executadas depois que ele for iniciado. As máquinas virtuais de domínio U são criadas, iniciadas e terminadas através do domínio 0 (CARISSIMI, 2008).

O domínio 0 consiste em uma máquina virtual única que executa um núcleo (*kernel*) *Linux* modificado e que possuí privilégios especiais para acessar os recursos físicos de entrada e saída e interagir com as demais máquinas virtuais (domínios U). Por ser um sistema operacional modificado, o domínio 0 possui os *drivers* de dispositivos da máquina física e dois *drivers* especiais para tratar as requisições de acesso à rede e ao disco efetuados pelas máquinas virtuais dos domínios U (CARISSIMI, 2008). A Figura 13 mostra o relacionamento entre o *hypervisor*, o domínio 0 e as demais máquinas virtuais.

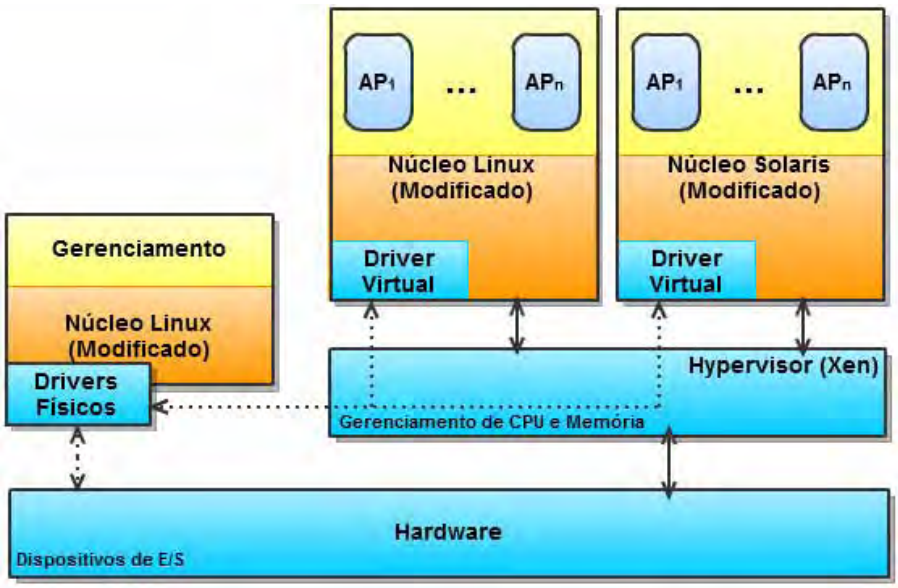

Figura 13 - Componentes do *Xen*: *hypervisor* e domínios. Adaptado de (CARISSIMI, 2008).

Como citado anteriormente, na primeira versão do *Xen* (lançada em 2003), o *hypervisor* foi implementado usando a técnica da paravirtualização, ou seja, sendo necessárias modificações nos sistemas operacionais hóspedes (domínios U) para torná-los conscientes do *hypervisor*. Essa decisão se justificava por questões de desempenho, mas limitou o emprego do *Xen* aos sistemas *Unix*, principalmente aqueles com filosofia de código aberto (CARISSIMI, 2008). A partir da 3ª versão, o *Xen* passou a oferecer virtualização completa, permitindo o uso de sistemas operacionais não modificados, como os da família Microsoft *Windows*. Entretanto, isso só é possível se o processador oferecer suporte de *hardware*, como as soluções Intel VT-x (INTEL, 2010) e AMD-V (AMD, 2010).

Para oferecer suporte tanto à paravirtualização quanto à virtualização completa, o *Xen* distingue os domínios U entre paravirtualizados (domínios U-PV) e virtualizados (domínios U-HVM, de *Hosted Virtual Machines*). Os domínios U-PV têm conhecimento de que não possuem acesso direto ao *hardware* e sabem da existência de outras máquinas virtuais. Os domínios U-HVM não têm essa consciência e nem mesmo reconhecem a existência de outras máquinas virtuais. Na prática, isso reflete no fato de que os domínios U-PV possuem *drivers* específicos para acesso a rede e a disco para interagirem com as suas contrapartidas no domínio 0. Em contraste, as máquinas dos domínios U-HVM não possuem tais *drivers* (já que os sistemas operacionais não foram modificados) e iniciam como um sistema convencional procurando executar a BIOS. O *Xen virtual firmware* simula a existência da BIOS executando todos os procedimentos esperados para o *boot* normal de um sistema operacional nativo. O compartilhamento do disco e as requisições de rede de um domínio U-HVM são feitos através de um *daemon* QEMU vinculado a cada instância U-HMV (O QEMU é um emulador em *software* de código livre). O *hardware* disponível para as máquinas virtuais do domínio U-HVM são aquelas oferecidas pelo QEMU (CARISSIMI, 2008).

3.7.3 *Virtual PC*

O *Virtual PC* é baseado na tecnologia que a Microsoft comprou, em 2003, da empresa Connectix, a qual foi pioneira no desenvolvimento de soluções de virtualização para computadores pessoais.

O *Virtual PC* consiste em uma máquina virtual para sistemas hospedeiros da família *Windows* que pode ser configurada para executar qualquer outro sistema operacional. Segundo a Microsoft, o principal objetivo do *Virtual PC* é o desenvolvimento e teste de *software* para múltiplas plataformas. Partindo deste princípio, o *Virtual PC* oferece mecanismos para interconectar logicamente as diferentes máquinas virtuais. Cada máquina virtual tem seu próprio endereço MAC e endereço IP. Além disso, o *Virtual PC* oferece um servidor de DHCP (*Dynamic Host Configuration Protocol*), um servidor NAT (*Network Address Translation*) e *switches* virtuais (CARISSIMI, 2008). Dessa forma, é possível construir cenários de rede usando máquinas virtuais. O *Virtual PC* está disponível para *download* gratuitamente, assim como um *white paper* que ensina a configurar as máquinas virtuais e um ambiente de rede.

3.7.4 *QEMU*

O QEMU é um emulador de *software* livre (quase todo LGPL, *Lesser GNU General Public License*, e GPL, exceto alguns *drivers*) para sistemas de PC completos. Além de emular um processador, o QEMU permite a emulação de todos os subsistemas necessários, como *hardware* de interligação de redes e de vídeo. Ele também permite a emulação de conceitos avançados, como sistemas de multiprocessamento simétrico (até 255 CPUs) e outras arquiteturas de processador, como ARM ou *PowerPC* (JONES, 2007). É um dos poucos hipervisores recursivos, ou seja, é possível chamar o QEMU a partir do próprio QEMU (TEIXEIRA, 2010). Com o QEMU não são necessárias alterações ou melhorias no sistema hospedeiro, pois ele utiliza intensivamente a tradução dinâmica como técnica para prover a virtualização. Além disso, ele suporta muitos sistemas operacionais convidados, incluindo *Linux*, *Solaris*, *Microsoft Windows*, DOS e BSD, emulando diversas plataformas de *hardware*, incluindo x86, x64 (AMD64/Intel 64), ARM, *Alpha*, *ETRAX CRIS*, MIPS, *MicroBlaze*, *PowerPC* e SPARC.

O *hypervisor* QEMU oferece dois modos de operação (TEIXEIRA, 2010):

- *Emulação total do sistema*: emula um sistema completo, incluindo processador e vários periféricos. Neste modo o emulador pode ser utilizado para executar diferentes sistemas operacionais;

- *Emulação no modo de usuário*: disponível apenas para o sistema *Linux*. Neste modo o emulador pode executar processos *Linux* compilados em diferentes plataformas (por exemplo, um programa compilado para um processador *x86* pode ser executado em um processador *PowerPC* e vice-versa).

O QEMU sempre foi um projeto ativo e seu desempenho sempre melhorava gradativamente com lançamentos de novas versões, porém, em 2005 algo radical aconteceu: foi lançado o KQEMU (*Kernel QEMU*, também chamado de *QEMU Acelerator*), um módulo de Kernel que transforma o QEMU *for Linux* num virtualizador similar ao *VMware*. Ele passa a repassar as chamadas ao invés de interpretar cada uma, como fazia até então. Mais do que isso, o desempenho também fica de quatro a cinco vezes melhor, também atingindo um patamar próximo ao do *VMware* (MORIMOTO, 2009b).

O KQEMU, ao contrário do QEMU, não possui código fonte aberto, mas é distribuído gratuitamente. Segundo publicado em (QEMU, 2010), usando o KQEMU o desempenho fica entre 50 e 100% do original. Isto se aplica ao desempenho bruto do processador. O acesso ao disco rígido e à memória RAM e também o desempenho do *driver* de vídeo virtual são bem mais baixos que o normal, o que faz com que, em aplicações práticas, você tenha de 25 a 50% do desempenho original da máquina. É menos do que o oferecido pelo *VMware*, porém mais que as versões antigas do QEMU.

Com o uso deste módulo, ao invés de interpretar cada uma das chamadas de sistema emitidas pelos processos convidados, o QEMU passa a executá-las diretamente sobre o sistema hospedeiro. O KQEMU permite associar os dispositivos de I/O e o endereçamento de memória do sistema convidado aos do sistema hospedeiro. Processos em execução sobre o núcleo convidado passam a executar diretamente no modo usuário do sistema hospedeiro. O modo núcleo do sistema convidado é utilizado apenas para virtualizar o processador e os periféricos (TEIXEIRA, 2010).

3.7.5 *KVM*

O KVM (*Kernel-based Virtual Machine*) é uma solução completa de virtualização para *Linux* em *hardware* que contenha extensões de virtualização (processadores VT-x ou AMD-V). É constituída por um módulo de *kernel* carregável, kvm.ko, que fornece a infra-estrutura de virtualização de núcleo e um módulo específico de processador, intel.ko kvm ou amd.ko kvm. O KVM suporta sistemas convidados *Windows*, *Linux* e *Unix*, podendo estes ser de 32 bits e 64 bits.

Esta solução é uma estrutura de virtualização inserida no *kernel* do *Linux*, desenvolvido com o objetivo de tornar o próprio núcleo um *hypervisor*, tendo sido implementado na forma de módulos carregáveis visando uma maior simplicidade (HABIB, 2008).

De forma geral, os processos em sistemas operacionais *Linux* apresentam dois modos de execução, uma aplicação executa em modo usuário até o momento em que se tornam necessários serviços mais privilegiados, como escrever dados no disco rígido, enviando assim um pedido para o modo *kernel*, que executa a operação. O KVM trata cada máquina virtual como sendo um processo padrão do gerenciador de processos do *Linux*, incluindo um terceiro modo, o modo convidado, o qual executa a partir máquina virtual (KIVITY et al., 2007).

O KVM não é, na verdade, considerado como um esquema de virtualização completo por si só, mas sim como parte de uma solução maior. Seu funcionamento consiste na utilização de uma interface localizada em /dev/kvm, para criar um espaço de endereçamento único para a VM, simulando a entrada e saída (I/O) de dados e mapeando a saída de vídeo para o hospedeiro. O único programa que se utiliza do KVM é o QEMU, desde a versão 0.10.0. A execução de entrada/saída de um sistema operacional convidado é fornecida com QEMU.

Uma vantagem do KVM é que, como ele faz parte do próprio *kernel*, pode se beneficiar das otimizações e avanços deste. As duas maiores desvantagens do KVM são: ele requer processadores mais novos, com extensões para virtualização, e um processo QEMU de espaço de usuário para fornecer virtualização de I/O.

3.7.6 *VirtualBox*

O *VirtualBox* (VIRTUALBOX, 2010) é um *software* de virtualização originalmente desenvolvido pela a empresa alemã Innotek até o início de 2008. Posteriormente, foi adquirida pela Sun[31], companhia que em 2010 tornou-se aquisição da Oracle Corporation[32]. Com isso, o até então conhecido Sun *VirtualBox* recebeu uma nova nomenclatura: *Oracle VM VirtualBox*.

O *VirtualBox* oferece amplo suporte a plataformas hospedeiras: *Microsoft Windows*, *Mac OS*, *Linux* e *Solaris*. Para sistemas convidados a lista é ainda maior: *Microsoft Windows*, *Linux*, *Solaris*, *OpenSolaris*, *FreeBSD*, *OpenBSD* e OS2 (SUN MICROSYSTEMS, 2010). Assim como outras ferramentas de virtualização as máquinas virtuais do *VirtualBox* podem ser inicializadas, pausadas e paradas.

Atualmente, a versão oficial do *VirtualBox* está sob uma licença proprietária para uso comercial. Para uso pessoal e educacional é utilizada a licença PUEL (*Personal Use and Educational License*) (PUEL, 2008). Uma alternativa à versão padrão é a *VirtualBox Open Source Edition – OSE*, um programa livre regido sob as regras GPL. Nesta segunda versão, duas funcionalidades são reprimidas: o suporte ao *Remote Desktop Protocol* - RDP[33] e o suporte a dispositivos USB.

O *VirtualBox*, foi em grande parte desenvolvido a partir do projeto QEMU + KQEMU e, em função disto, utiliza como mecanismo de virtualização a recompilação dinâmica de instruções, método baseado na emulação por tradução binária do QEMU. Da mesma forma que o KQEMU, o *VirtualBox* executa a maior parte dos códigos de usuários e de *kernel* diretamente sobre o processador utilizando seu VMM . As semelhanças entre o *VirtualBox* e o QEMU também se estendem ao fato de que todo o modelo de emulação de dispositivos periféricos utilizado no primeiro também é baseado no modelo de dispositivos virtuais do segundo (CRUZ, 2008).

[31] http://br.sun.com
[32] http://www.oracle.com
[33] RDP consiste em um protocolo proprietário desenvolvido pela Microsoft que visa fornecer a um usuário uma interface gráfica para outro computador.

O *VirtualBox* tem um *software* especial, conhecido como Adicionais para Convidados, que pode ser instalado nos sistemas operacionais virtualizados para melhorar o desempenho e proporcionar melhor integração entre sistema hospedeiro e sistema convidado (VIRTUALBOX, 2010). Entre os recursos fornecidos por esses adicionais estão integração do ponteiro do mouse e resoluções de tela (permitindo o redimensionamento da tela do sistema convidado de acordo com a do sistema hospedeiro).

O *VirtualBox* possui versões para processadores de 64 bits e possibilita o uso do "*hardware assisted virtualization*" se aproveitando das instruções especiais encontradas nos processadores mais recentes, tais como Intel VT-X e AMD-V, melhorando o desempenho da virtualização.

Outra característica interessante do *VirtualBox* é que as definições de configuração de máquinas virtuais são armazenadas em arquivos XML[34] e são totalmente independentes das máquinas locais, facilitando a portabilidade (VIRTUALBOX, 2010).

Tal como muitos outras soluções de virtualização, para facilitar a troca de dados entre os hospedeiros e máquina virtual, o *VirtualBox* permite a declaração de diretórios do hospedeiro como "pastas compartilhadas", que podem ser acessados de dentro das máquinas virtuais.

3.8 Considerações Finais

A virtualização, sem dúvidas, se tornou em uma das novas tendências de revolução na área de TI como a solução para empresas que desejam diminuir custos sejam eles com equipamentos, recursos naturais (como a energia elétrica) ou pessoal capacitado, já que um número menor de equipamentos requer menos mão de obra para gerenciá-la (BARUCHI, 2008).

Um uso comum da virtualização é na consolidação de servidores, isso é, permitir que vários deles executem simultaneamente sobre um único *hardware* físico, porém isolados, ou seja, cada um em sua própria máquina virtual. Como cada máquina virtual é um ambiente

[34] Linguagens de marcação para necessidades especiais.

isolado, completo, os servidores podem ser de sistemas operacionais diferentes e, um ocasional comprometimento de um não afeta os demais. Além disso, a consolidação de servidores reduz investimentos em aquisição de equipamentos, e com infra-estrutura física como espaço, energia, cabeamento, refrigeração e custos de gerenciamento e manutenção. Isso é especialmente interessante em *data centers* (CARISSIMI, 2008).

Além de utilizada em servidores, a virtualização também pode ser empregada com sucesso em diversas outras situações como ambientes de desenvolvimento e teste de produtos, laboratórios de treinamento de cursos de redes e de sistemas operacionais, criação de clusters ou grades computacionais virtuais e servir de base para implantação de mecanismos de segurança (*honeypots*) (CARISSIMI, 2008).

Como a virtualização consiste basicamente em pôr uma camada de *software* a mais em um sistema computacional, existem perdas consideráveis de desempenho quando se virtualiza determinados serviços. Estudos feitos pela *VMware* e pela *XenSource* apontam para uma queda de desempenho, em geral, entre 2% e 10%, com algumas situações impondo perdas maiores (CARISSIMI, 2008). Cabe ressaltar que esses resultados foram obtidos usando *benchmarks* genéricos. Porém, a tendência é que essas perdas venham a diminuir cada vez mais ao longo do tempo, já que tanto os desenvolvedores de maquinas virtuais quanto as empresas de *hardware* vêm investindo esforços em melhorias relacionadas à virtualização.

4. A ARQUITETURA DO MODELO WSE-OS

4.1 Considerações Iniciais

Este Capítulo apresenta as características do modelo gratuito de configuração de ambiente computacional, denominado WSE-OS (*Wireless Sharing Enviroment – Operating Syste*ms), que visa centralizar o gerenciamento do ambiente em um ponto central, proporcionando maior facilidade da tarefa de gerenciamento, flexibilidade para os usuários e segurança dos dados. Também serão apresentadas sua arquitetura e fluxo de execução, além de uma breve contextualização de gerenciamento centralizado de computadores ilustrando características e limitações das soluções atuais. Posteriormente serão discutidos os módulos que compõem este modelo, identificando qual a função de cada módulo no funcionamento da solução WSE-OS.

4.2 Gerenciamento Centralizado de Computadores

Atualmente, grande parte dos ambientes computacionais, sejam eles corporativos, educacionais ou domésticos, apresentam um alto nível de interconexão entre seus elementos. Na prática, essa conectividade faz com que os administradores de sistemas sejam responsáveis por gerenciar um conjunto grande e heterogêneo de computadores, cada um dotado de sistemas operacionais diferentes e com conjuntos de aplicações também diferentes ente si, dificultando a tarefa do administrador.

De forma geral, soluções que oferecem recursos para gerenciamento de computadores em rede adotam mecanismos para concentrar as operações administrativas em um ponto central. Desta forma, todas as políticas e tarefas de gerenciamento são acionadas a partir de um único ponto para serem refletidas nos demais nós gerenciados da rede (CHERVENAK et al., 2000). A possibilidade de gerenciar computadores de um ponto central se reflete na arquitetura de rede adotada por estas soluções, sendo que na maioria destas arquiteturas existe um computador atuando como servidor com serviços de gerenciamento e diversos clientes conectados a ele, enviando informações de *status* e recebendo informações e

comandos de configuração (CRUZ, 2008), como mostra a Figura 14.

Figura 14 - Arquitetura de rede de computadores utilizando um servidor de gerenciamento e clientes gerenciados na rede.

Dentre os serviços de gerenciamento oferecidos pelas soluções de gerenciamento centralizado, destacam-se (CRUZ, 2008):

- Definição centralizada dos parâmetros de configuração de *software* para todos os computadores;
- Distribuição, instalação e configuração centralizada de pacotes de *software* aplicativo;
- Definição centralizada de políticas de uso e permissões de acesso aos recursos dos computadores;
- Definições centralizadas dos serviços de segurança, proteção de dados e cópias de segurança de informações;
- Geração centralizada de inventários de *hardware* e *software*.

Os serviços listados anteriormente visam à automação de tarefas repetitivas e relativamente simples de configuração de *software*. De maneira geral, os modelos de gerenciamento centralizado de computadores podem ser divididos em dois grandes grupos de soluções:

- Grupo 1 - Execução Centralizada: Soluções que têm como principal característica a

execução centralizada dos ambientes de trabalho dos clientes no servidor da rede;

- Grupo 2 - Configuração Centralizada: Soluções que têm como principal característica a centralização das informações de configuração de *software* dos computadores clientes.

No Grupo 1 se encaixam soluções como gerenciamento através de *thin clients*, baseado em comunicação remota TCSC (*Thin Client Server Computing*), e infra-estrutura virtual de *desktops* (*Virtual Desktop Infra-structures*). Nos dois casos, a capacidade de *hardware* do servidor, responsável por hospedar o processamento dos computadores clientes, consiste no principal gargalo operacional para se obter uma solução que ofereça um custo-benefício linear em função do número de computadores. Por exemplo, o custo para se criar uma infra-estrutura para gerenciar quarenta computadores pode não ser compatível com as necessidades da organização, sendo o mesmo válido para projetos com mais de centenas de computadores, já que o número de computadores a ser suportado poderia exigir investimentos em equipamentos de alto desempenho, como os encontrados em *data centers*, para se criar a central de servidores (CRUZ, 2008).

No Grupo 2 encontram-se as soluções que aplicam seus mecanismos de gerenciamento centralizado em camada de aplicativo. Como exemplo tem-se *scripts* que automatizam atividades principais de gerenciamento, e também soluções que funcionam na camada de sistema operacional, usando o conceito de Imagem de Sistema Única (ISU) (ZHOU, ZHANG & XIE. 2006). No segundo caso, o principal mecanismo de gerenciamento emprega uma imagem de um disco rígido, que nada mais é que um arquivo binário representando os volumes lógicos de um disco, suas partições e dados armazenados, gerada a partir de um computador cuja configuração de *software* é considerada como sendo o padrão a ser replicado nos demais computadores da rede (CRUZ, 2008).

Soluções que se baseiam em gerenciamento através de Imagens de Sistema Únicas ou as baseadas em *scripts* em camada de aplicativos possuem como limitação o forte acoplamento entre *hardware* e *software* dos terminais, impossibilitando o gerenciamento de um ambiente que possua uma configuração heterogênea de dispositivos com apenas uma ISU ou um único *script* padrão de configuração (CRUZ, 2008). Tais soluções são mais aplicáveis a

redes homogêneas, ou seja, que apresentam a mesma configuração de *hardware* em suas máquinas.

As inúmeras possíveis configurações em decorrência da combinação de diferentes *hardware* e *software* podem gerar uma complexidade de gerenciamento tão grande a ponto de inviabilizar o uso de soluções baseadas em ISU ou *script* único, já que as tarefas dos gerentes de ambientes acabam sendo individualizadas a cada conjunto diferente de *hardware* do ambiente (SIRER et al., 1998).

Um tipo emergente de solução de gerenciamento com execução centralizada utiliza uma arquitetura que combina mecanismos de virtualização e acesso remoto. Tais soluções são conhecidas como *Virtual Desktop Infrastructure* (VDI) e, como já visto anteriormente, se encaixam no grupo de Execução Centralizada. Neste modelo arquitetônico os sistemas operacionais clientes são executados em máquinas virtuais (VMs) num servidor, permitindo que as aplicações dos usuários rodem isoladamente e ao mesmo tempo compartilhem recursos de *hardware* como CPU, memória, disco e rede. O protocolo de conexão entre o usuário e o *desktop* que está hospedado no servidor pode ser o RDP, VNC e ICA[35].

O VDI se ajusta melhor para usuários que precisam de acesso a seu ambiente de trabalho de qualquer lugar, inclusive a partir de diferentes computadores, assim como clientes corporativos com uma estratégia de estações de trabalho centralizada para funcionários de escritório. Usuários que utilizam aplicações gráficas intensivas ou aplicações com requisitos de *streaming* vídeo e áudio têm boas chances de não utilizar o VDI, em função das altas taxas de transmissão exigidas.

O *VMware Virtual Desktop Infrastructure* (VMWARE, 2007) é um exemplo dessa solução. Ele cria uma infra-estrutura de *desktops* virtualizados onde a execução e gerenciamento de dados ocorre em um único servidor. Desta forma, terminais acessam *desktops* virtuais através dos servidores de autenticação e alocação de máquinas virtuais nos quais estão sendo gerenciados e configurados remotamente.

[35] Acrônimo para *Independent Computing Architecture*. Consiste em um protocolo proprietário para um sistema de servidor de aplicação, projetado pela Citrix Systems.

4.3 O Modelo WSE-OS

O WSE-OS é um modelo inovador para gerenciamento de ambientes computacionais que visa facilitar a tarefa dos administradores de ambientes centralizando as atividades administrativas em um único ponto, oferecendo a coordenação de instanciações de sistemas operacionais através de invocações remotas por um canal sem fio entre clientes e servidor.

O sistema integra e sincroniza elementos de *hardware* e *software* de clientes de modo a controlar os recursos de rede. Baseado no modelo arquitetural cliente-servidor com execução centralizada, o WSE-OS é estruturado no conceito de sistemas VDI e utiliza a comunicação entre objetos distribuídos para oferecer um ambiente de execução completo para os usuários do sistema. A Figura 15 ilustra a arquitetura do sistema WSE-OS.

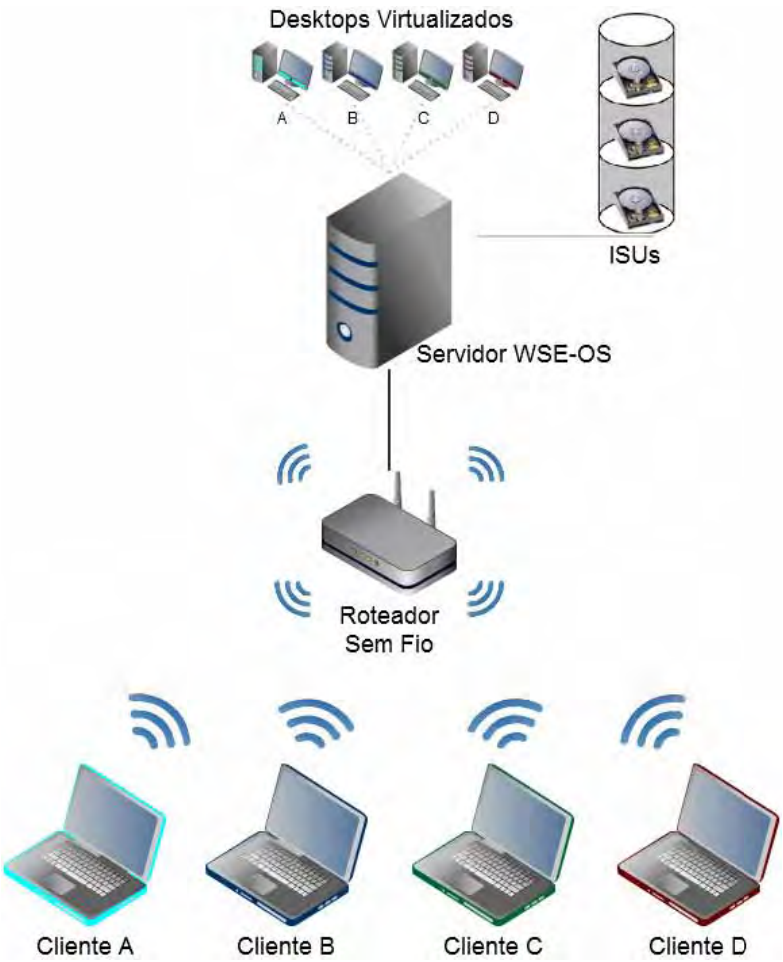

Figura 15 - Arquitetura do sistema WSE-OS.

A arquitetura do modelo visa à flexibilidade e à facilidade de conexão na qual associações entre as diversas estações clientes e o servidor são constantemente criadas e destruídas (Execução Centralizada). Ao contrário do que as atuais soluções de gerenciamento centralizado de imagens, como o FlexLab (AGUIAR et al., 2009), *Citrix Provisioning Server for Desktops* e o *Ardence Desktop Streaming* oferecem, o WSE-OS é especificamente projetado para utilizar rede sem fio para transmissão de dados, propondo uma alternativa de conexão não cabeada entre servidor WSE-OS e as estações clientes a fim de compartilhar a execução de imagens de sistemas operacionais.

Baseado no conceito de *boot* remoto e do projeto LTSP (*Linux Terminal Server Project*), uma distribuição *Linux* destinada a ser carregada pelos terminais que não possuem poder de processamento ou memória RAM suficiente para rodarem distribuições de SO mais atuais com bom desempenho (MORIMOTO, 2008), o WSE-OS possui características de terminais leves (*thin clients*) onde há a necessidade de recursos mínimos de *hardware* no cliente para que este possa apresentar as imagens de tela de um sistema operacional.

Por tirar proveito de toda execução centralizada provida pelo sistema de acesso remoto, periféricos como discos rígidos locais, CD-ROM, processadores de alta frequência e memórias de grande capacidade são abstraídos totalmente pelo *middleware* de comunicação refletindo diretamente no consumo de energia, no baixo custo de *hardware* e administração dos clientes WSE-OS.

Os clientes podem ser iniciados pela instanciação de um sistema operacional no servidor multiprocessado. Este, por sua vez, mantém o isolamento de cada instância de execução, ou seja, o sistema de um usuário jamais interferirá no de outro. O responsável pelo pedido de instanciação é o *middleware* WSE-OS de comunicação remota, que deve estar em alguma mídia de armazenamento secundária como *pen-drive* e CD-ROM, no computador cliente. Os benefícios decorrentes do uso da solução WSE-OS são:

- Proteção Contra Perda de Dados: pelo fato de os dados estarem centralizados, é mais simples protegê-los contra perdas e realizar *backups*;
- Redução de Ameaças de Roubo: com os dados centralizados, as regras de proteção dos dados ficam mais eficientes e fáceis de serem implementadas;

- Atualização de SO e Aplicativos: *patches* de segurança e atualizações de antivírus, sistema operacional, aplicações, entre outros, são mais fáceis de se manter atualizados quando as manutenções são realizadas em um único lugar (servidor), em vez de fazê-las para cada PC individual distribuído;
- Redução de Tempo com Configurações: a configuração do *desktop* virtual leva aproximadamente 30 minutos contra horas para um computador normal;
- Economia de Energia: *desktop* virtual consome menos eletricidade;
- Mobilidade: *desktops* acessados via rede sem fio com segurança possibilitam trabalhar de/em qualquer lugar das empresas ou organizações;
- Isolamento: como os sistemas operacionais de cada usuário são isolados, o eventual travamento de um SO não influencia no funcionamento dos outros.

A Figura 16 auxilia na visualização dos benefícios listados.

Figura 16 - Diagrama do WSE-OS. Adaptado de (VMWARE, 2011c).

A execução do *middleware* nos terminais assegura que toda a comunicação posterior será realizada por meio sem fio, garantindo uma maior flexibilidade do sistema, pois não obriga que o processo de *boot* seja realizado por protocolos como o PXE, padrão aberto adotado pela indústria para permitir que clientes de uma rede corporativa possam carregar automaticamente através da rede imagens de sistemas operacionais (HENRY, 2000). A Figura

17 ilustra a arquitetura de camada do cliente e servidor WSE-OS, que é detalhada na seção 4.4.

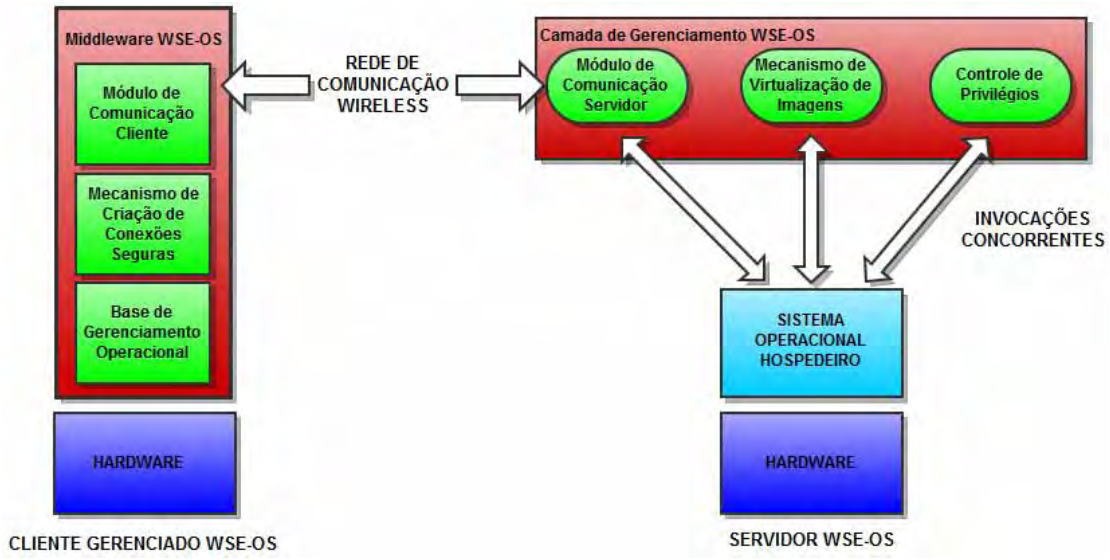

Figura 17 - Arquitetura de camada do WSE-OS.

O funcionamento do modelo WSE-OS é orientado a comunicação por fluxo, uma vez que o SO utilizado pelo cliente através do mecanismo de acesso remoto se torna indisponível sem a conexão TCP. O diagrama da Figura 18 apresenta o fluxo de execução do sistema WSE-OS.

O WSE-OS utiliza o modelo cliente-servidor baseado em execução centralizada através de comunicação remota TCSC (*Thin Client Server Computing*), no entanto com algumas alterações visando uma melhora no desempenho. A arquitetura possui em cada *middleware* cliente um módulo *cache* junto com um sistema de transferência de dados diferencial, conforme explicado na seção 2.4.4. O sistema apresenta simplicidade e flexibilidade na associação e dissociação de novos terminais clientes por ser estruturado com conexões sem fio, além de, pelo fato de utilizar virtualização, não possuir acoplamento entre *hardware* e *software* de um computador como acontece nos casos de gerenciamento de configuração centralizada.

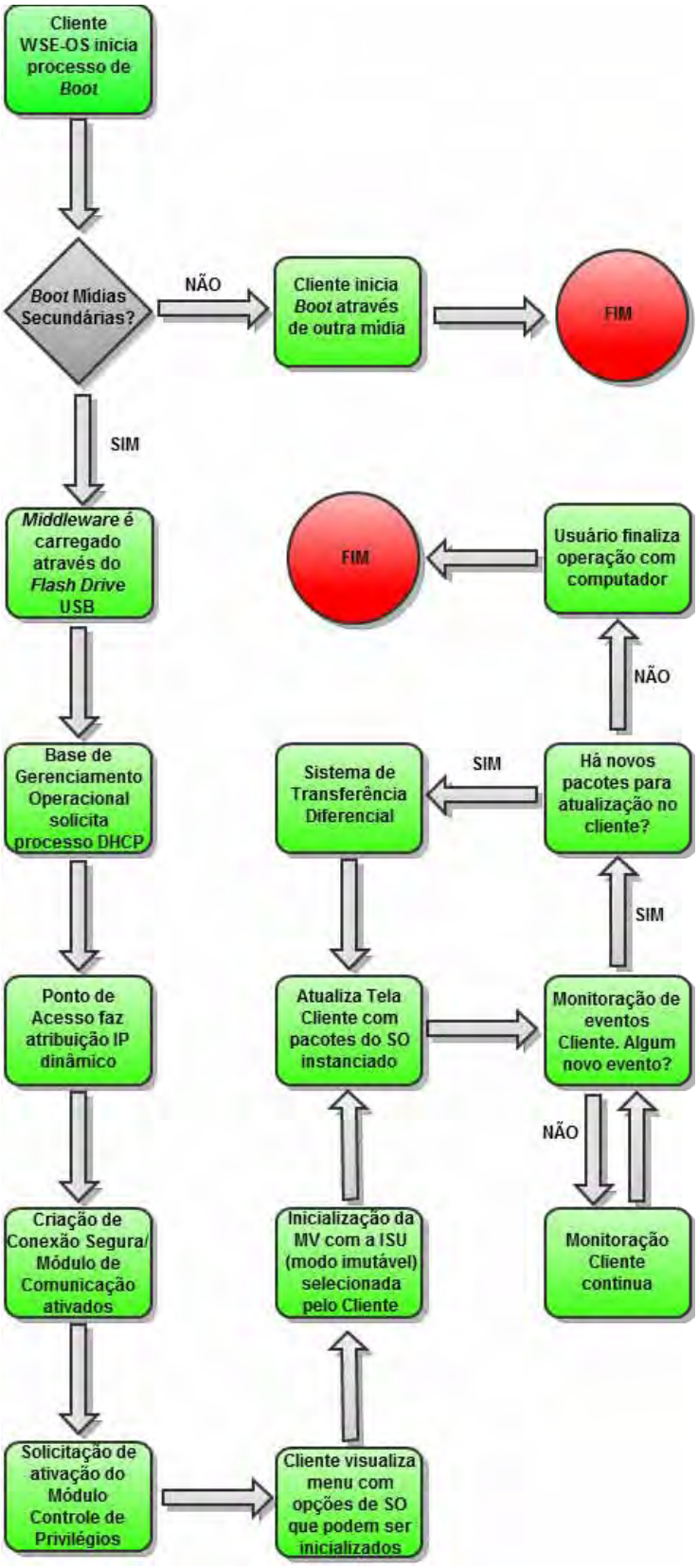

Figura 18 - Fluxo de execução do sistema WSE-OS.

Através do fluxo apresentado pela Figura 18 é possível verificar que o *middleware* é um elemento indispensável ao modelo WSE-OS, permitindo que os clientes possam ter acesso ao servidor e que, a partir daí, possam interagir com seus respectivos SOs sendo executados remotamente. Além disso, ele é o responsável pela solicitação de ativação do Módulo de Controle de Privilégios, presente no servidor, que construirá uma interface gráfica para que os usuários possam escolher qual SO deverá ser iniciado. O fluxo também mostra que apenas são trocados dados entre cliente e servidor caso algum evento seja realizado pelo primeiro, caso contrário a conexão continua ativa, porém não consome banda de rede, evitando congestionamentos desnecessários.

4.4 Componentes do WSE-OS

Nesta seção serão caracterizados os elementos fundamentais que compõem o projeto de Ambiente de Compartilhamento Sem Fio para Sistemas Operacionais. O WSE-OS é composto por dois grandes componentes: 1. *Middleware* de Comunicação Remota sem fio; 2. Módulo de Gerenciamento WSE-OS (objetivo deste trabalho). Além destes componentes têm-se a rede sobre a qual todo o ambiente é estruturado.

4.4.1 *Middleware* de Comunicação Remota sem Fio WSE-OS

O *Middleware* de Comunicação Remota sem Fio WSE-OS é um sistema localizado entre o *hardware* do cliente gerenciado e o módulo de gerenciamento WSE-OS que oferece a instanciação de um sistema operacional. Este mediador é composto por três módulos hierarquicamente dependentes funcionalmente sendo responsável pela transmissão e recepção de dados do servidor, conforme a Figura 17.

A primeira camada do *middleware* é composta por uma base operacional responsável pelo gerenciamento de recursos deste cliente, controlando processos, memória, sistema de arquivo e periféricos. A segunda camada tem o objetivo de criar conexão segura por fluxo de dados com o servidor, utilizando mecanismo troca de chaves e criptografia. A terceira

camada, o módulo de comunicação cliente, garante a transmissão e recepção de dados entre os processos do cliente com os processos do servidor WSE-OS.

4.4.2 Camada de Gerenciamento WSE-OS

A Camada de Gerenciamento do WSE-OS consiste em um sistema localizado no servidor WSE-OS que reúne ferramentas de virtualização, acesso remoto, ambiente de trabalho remoto e sistema de arquivos, visando oferecer uma solução para que um cliente remoto seja capaz de interagir com uma instanciação de um sistema operacional. Como pode ser visto na Figura 17, esta camada é composta por três módulos (Módulo de Comunicação Servidor, Mecanismo de Virtualização de Imagens e Controle de Privilégios) independentes funcionalmente entre si, mas que quando executando simultaneamente são capazes de configurar o servidor para que este seja capaz de fornecer acesso, pelos clientes, a um sistema operacional virtualizado. O Capítulo 5 apresenta de forma detalhada cada característica da Camada de Gerenciamento WSE-OS.

4.4.3 Rede de comunicação

O Ambiente de Compartilhamento de Dados entre servidor e clientes é baseado na modalidade infra-estrutura WLAN (*Wireless Local Area Network*), em que os dispositivos-clientes comunicam-se sem fio diretamente com o ponto de acesso e este, por consequência, troca mensagens por conexão física com o servidor WSE-OS.

O padrão de comunicação sem fio utilizado é o 802.11n, já que possui velocidade de transmissão superior aos seus concorrentes (802.11b, 802.11a e 802.11g), além de melhorias com relação à latência, ao alcance e à confiabilidade de transmissão. A velocidade nominal do padrão 802.11n é de 300 megabits, contra 54 megabits do 802.11g (padrão mais popular atualmente). Além disso, o padrão 802.11n consegue um alcance de sinal aproximadamente duas vezes maior que do 802.11g e um nível mais elevado de confiabilidade, ambos em função do uso de múltiplos fluxos de transmissão.

Com relação à transmissão dos dados, o principal entrave de uma rede sem fio é que o meio de transmissão (o ar) é compartilhado por todos os dispositivos conectados ao ponto de acesso, pelo fato de o sinal ser simplesmente irradiado em todas as direções. Isso significa que qualquer pessoa, usando um computador com uma antena suficientemente sensível, pode captar o sinal da rede e, se nenhuma precaução for tomada, ter acesso aos dados que trafegam pela rede ou até mesmo ingressar na mesma. Assim sendo, a segurança no contexto de um ambiente sem fio é uma questão delicada, pois a interceptação dos dados acaba sendo facilitada para entidades mal-intencionadas.

Em função disso, este projeto utiliza para segurança o padrão WPA2-PSK, um sistema que visa garantir confidencialidade dos dados e controle de acesso. O WPA2 é um protocolo de comunicação sem fio que faz uso do algoritmo AES, um sistema de criptografia utilizado pelo governo dos EUA e que é baseado no uso de chaves temporais de 128 a 256 bits, garantindo um nível elevadíssimo de segurança. Em conjunto com o AES é utilizado um sistema de autenticação denominado *Pre-Shared Key* (PSK), através do qual o ponto de acesso verifica se os usuários possuem a chave mestra da rede, sendo esta uma chave pré-compartilhada.

Desta forma, com o uso do padrão de transmissão 802.11n associado a mecanismos de segurança do sistema WPA-PSK, o WSE-OS é capaz de oferecer uma rede com altas taxas de transmissão (300 megabits por segundo), grande alcance (comparado aos protocolos concorrentes ao 802.11n) e confidencialidade assegurada.

4.5 Considerações Finais

O WSE-OS consiste em uma solução gratuita baseada no modelo VDI que é capaz de oferecer a clientes remotos um *desktop* completo através de um ambiente de comunicação sem fio, além de oferecer abstração de *hardware* aos clientes, tornando o gerenciamento mais flexível e independente de conexões físicas.

O modelo WSE-OS possui duas linhas de pesquisa e implementação. A primeira está focada na estruturação do *middleware* de comunicação remota, um mediador que deve estar presente em todos os clientes WSE-OS para a troca de dados com o servidor de desktops. A

segunda linha tem por objetivo a elaboração da Camada de Gerenciamento WSE-OS. Esta camada estará situada no servidor do sistema e será responsável pela coordenação de instanciações de sistemas operacionais para os clientes. Este trabalho aborda especificamente esta segunda linha de pesquisa, a da elaboração da Camada de Gerenciamento WSE-OS.

Com a associação de técnicas de virtualização e sistema de acesso remoto seguro, o servidor WSE-OS é capaz de oferecer aos clientes um ambiente de trabalho completo de forma otimizada e segura, por se utilizar de ferramentas especificamente elaboradas para redes com baixas taxas de transmissão e que podem ter o fluxo de comunicação interceptado por entidades mal-intencionadas.

Por ser um modelo baseado em execução centralizada, a tarefa de gerenciamento de ambientes computacionais tem sua complexidade reduzida, em função de ter todas as atividades de manutenção centralizadas em um único ponto do ambiente (o servidor). Além disso, os administradores de redes podem controlar eficientemente os privilégios de cada usuário através de uma interface que respeita as regras configuradas no sistema operacional do servidor.

5. A CAMADA DE GERENCIAMENTO WSE-OS

5.1 Considerações Iniciais

Neste Capítulo são apresentados os elementos que compõem a arquitetura da Camada de Gerenciamento, assim como as funções de cada elemento dentro da solução WSE-OS. Este Capítulo também apresenta uma análise sobre as ferramentas utilizadas em cada módulo da Camada de Gerenciamento, justificando as razões pelas quais foram escolhidas para a solução. Por fim, o Capítulo apresenta o funcionamento da Camada de Gerenciamento e suas características de execução.

5.2 Estrutura da Camada de Gerenciamento

Como já apresentado na seção 4.4.2, a Camada de Gerenciamento WSE-OS é composta por três módulos: Módulo de Comunicação Servidor, Mecanismo de Virtualização e Módulo de Controle de Privilégios.

O Módulo de Comunicação Servidor é responsável por manter o canal de troca de dados entre servidor e cliente. Este módulo é baseado na comunicação entre objetos distribuídos realizada por meio de invocações a métodos remotos (RMI – *Remote Method Invocation*) (CREPALDI, 2011) e, em um nível inferior, possui um mecanismo de criação de canal de comunicação seguro entre processos cliente e servidor. Tal estruturação é executada sobre o sistema operacional hospedado no Servidor WSE-OS, que é responsável não só por manter este módulo, mas também os demais módulos, como ilustrado na Figura 19.

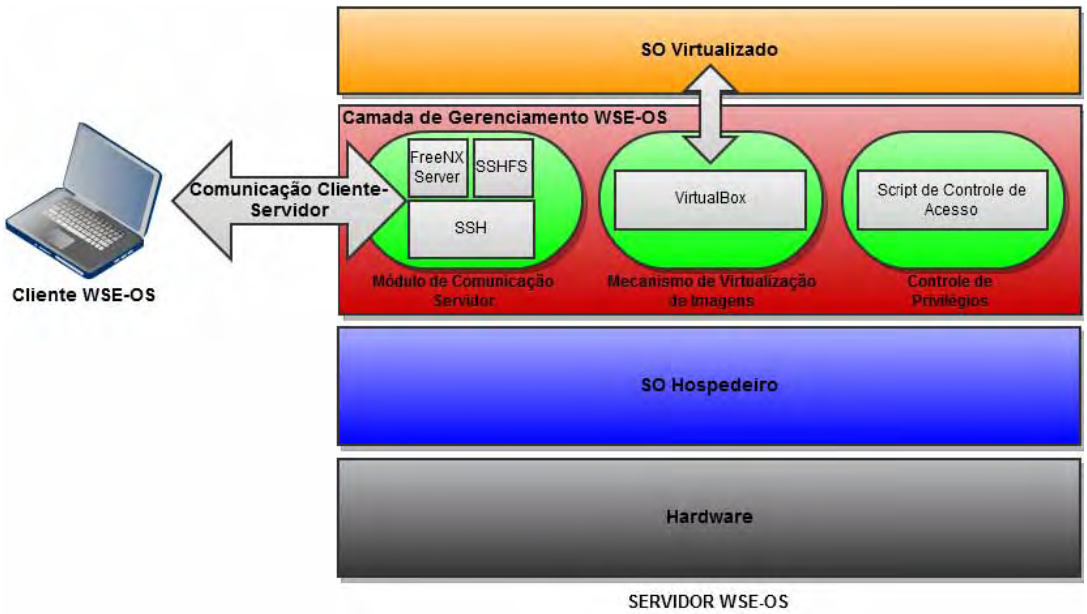

Figura 19 - Arquitetura detalhada do Camada de Gerenciamento WSE-OS

O mecanismo de criação de conexões seguras, que é o alicerce do Módulo de Comunicação Servidor, é baseado no protocolo de acesso remoto SSH (*Secure Shell*). O SSH é uma ferramenta que permite a criação das conexões entre computadores com confidencialidade e integridade dos dados através de técnicas de criptografia e túnel orientado a fluxo TCP (CREPALDI, 2011). A utilização desse sistema de comunicação remota tem o objetivo de encapsular o protocolo de comunicação servidor fornecendo transmissão segura já que os dados trafegam em uma rede sem fio, onde o sinal pode ser captado por uma entidade mal-intencionada.

Outra ferramenta que compõe o Módulo de Comunicação Servidor é o *FreeNX Server*. Esta é responsável pela troca de informações entre os processos cliente-servidor feita através da invocação a métodos remotos baseados em eventos. O *FreeNX Server* capacita o servidor a ser um provedor de Ambientes Operacionais independentes para cada cliente, que por sua vez acaba se tornando um *thin client*, ou seja, o módulo RMI baseado em eventos recebe as ações de periféricos do *middleware* localizado no cliente, processando tais eventos e retornando informações referentes à interface que deve ser apresentada no monitor do usuário.

A última ferramenta integrante do Módulo de Comunicação Servidor é o SSHFS (*SSH FileSystem* ou *Secure SHell FileSystem*) (HOSKINS, 2006), um sistema de arquivos remotos

que, neste projeto, tem a finalidade de permitir que os dispositivos removíveis conectados nas estações de trabalho remotas possam ser acessíveis pelos usuários através dos sistemas operacionais virtualizados sobre o servidor WSE-OS.

O Módulo de Virtualização é o responsável por manter em execução cada sistema operacional dos clientes. Ele carrega a ISU escolhida pelo usuário mantendo execução de todos os sistemas de forma independente e isolada, garantindo a integridade do ambiente.

Por fim, o Módulo de Controle de Privilégios consiste na interface gráfica do sistema e realiza a mediação entre os dois módulos anteriores (Módulo de Comunicação Servidor e Módulo de Virtualização). Ele entra em execução assim que se estabelece a comunicação cliente-servidor e é o responsável por colocar em execução a ferramenta de virtualização com a ISU escolhida pelo usuário. Além disso, foi uma preocupação deste trabalho apresentar uma interface de controle de acesso intuitiva, facilitando o uso do ambiente para usuários com menor experiência, e ao mesmo tempo funcional, permitindo que os gerenciadores de TI controlem de forma eficiente os privilégios de cada cliente.

A Arquitetura da Camada de Gerenciamento WSE-OS está projetada de maneira modular, o que torna possível, no futuro, a troca das ferramentas de cada módulo por tecnologias mais atuais que possuam melhores desempenhos ou que se adéquem melhor a situações específicas que apresentem características diferentes do ambiente previsto por este projeto, podendo em função disto, terem seus desempenhos melhorados com alterações na Camada de Gerenciamento. As seções seguintes apresentam de forma detalhada cada módulo da Camada de Gerenciamento WSE-OS e a base operacional utilizada pela solução.

5.3 Módulo de Comunicação Servidor WSE-OS

No Módulo de Comunicação Servidor, o mecanismo para criação de conexões seguras é baseado no sistema de acesso remoto *Secure Shell* (*OpenSSH* versão 1.2.12) (YLONEN, 2006a), como apresentado anteriormente. O SSH atende todas as necessidades para garantir confidencialidade e integridade nas transmissões vulneráveis à interceptação, sendo

considerado um protocolo padrão de transmissão segura com publicação pela IETF (*Internet Engeneering Task Force*) (CREPALDI, 2011).

Num nível mais acima do Módulo de Comunicação Servidor está a ferramenta responsável por oferecer a interface gráfica aos usuários. Para isso, será utilizada a aplicação *FreeNX Server* (versão 0.7.3), por requisitar menos banda de rede, se adequando ao meio de comunicação sem fio e internet, onde as taxas de transmissão são inferiores às redes locais cabeadas. A Figura 20 ilustra o Módulo de Comunicação Servidor, onde o SSH encapsulará as mensagens do *FreeNX* em um túnel criptografado utilizando o serviço SCP[36] (*Secure Channel Protocol*) (YLONEN, 2006b).

O *FreeNX Server* possui um sistema diferencial de transmissão, enviando seletivamente os dados de som, vídeo, imagem e texto, utilizando os algoritmos *wav*[37]*, mpeg*[38]*, jpg/png e zlib*, respectivamente. Além disso, ele possui dois *proxys*: o "*nxproxy*", que cria um *cache* para armazenamento de dados já transferidos, para posterior reutilização, e o "*nxagent*", que elimina os pacotes de resposta ("*roundtrips*"). Tais características fazem do *FreeNX Server* uma ferramenta adequada para conexões com taxas limitadas de transmissão de dados, como redes sem fio, na qual a solução WSE-OS é estruturada. Outros sistemas para acesso a *desktop* remoto, como por exemplo, o *RealVNC*, *TightVNC* e *X Window System* não possuem todos os requisitos exigidos pelo ambiente WSE-OS (otimização de compressão de dados) e, por isso, não são adequados para o Módulo de Comunicação Servidor.

A terceira, e última, ferramenta que compõe o Módulo de Comunicação Servidor é o SSHFS em sua versão 2.2. Esta tem por finalidade permitir acesso às mídias removíveis dos terminais remotos. No modelo de gerenciamento centralizado proposto pelo WSE-OS, os usuários possuem seus sistemas operacionais em execução em um servidor remoto, sendo que todo o processamento dos primeiros é realizado pelo segundo. Em função disso, para que os clientes remotos possam acessar seus dispositivos removíveis, estes devem estar acessíveis para o servidor. E a partir desta máquina que devem surgir as requisições de montagem de dispositivos, que estão presentes nos terminais remotos. Diante dessa

[36] Protocolo de rede que fornece um meio seguro para transferência de arquivos entre computadores remotos.
[37] Formato padrão de arquivo de áudio da Microsoft e IBM.
[38] Padrão de compressão de vídeo para aplicações multimídia.

necessidade, o Módulo de Comunicação Servidor se utiliza do SSHFS para permitir que o servidor WSE-OS tenha acesso às mídias removíveis dos clientes, para que desta forma os usuários também possam ter acesso a esses recursos.

O SSHFS permite que um sistema de arquivos remoto seja disponibilizado em uma máquina local, ou seja, permite que um diretório que esteja em um computador fisicamente distante esteja disponível para uma máquina local, como se fosse um diretório nativo do sistema de arquivos. Para isto, o SSHFS se utiliza de um protocolo denominado de SFTP (*Secure File Transfer Protocol*) (YLONEN, 2006b), uma extensão do SSH, para prover um método seguro de transferência de arquivos (HOSKINS, 2006), característica esta que faz do SSHFS uma ferramenta indispensável para ambientes em que a transmissão de dados se dá por meio sem fio, como no WSE-OS, onde os fluxos de dados podem ser interceptados.

Embora o SFTP permita transferência de arquivos e diretórios, ele sozinho não é capaz de estabelecer um ponto de montagem local do sistema de arquivos remoto, por isso o uso do SSHFS. Ele é construído sobre o projeto FUSE (*Filesystem in USErspace*), um mecanismo que provê a implementação de sistemas de arquivos em espaço de usuário em ambientes POSIX (REAL, 2009). Em função disso, o SSHFS é capaz de se conectar ao sistema remoto e fazer todas as operações necessárias para prover a aparência de uma interface de sistema de arquivos normal de arquivos remotos. Com ele, diretórios remotos podem ser montados localmente da mesma forma de outros volumes (como CDs, DVDs, *pen drives* e discos compartilhados), permitindo que os dispositivos removíveis dos clientes WSE-OS possam ser acessados do lado do servidor, podendo os usuários trabalhar com os arquivos e diretórios remotos como se estivessem em um volume local.

O SSHFS trabalha em conjunto com o *VirtualBox* para permitir que os usuários tenham acesso aos dispositivos removíveis através do SO virtualizado. Desta forma, o primeiro oferece acesso, por parte do servidor, aos dispositivos nos clientes e o segundo possibilita que o usuário consiga interagir com os arquivos, que estão em um diretório do sistema operacional hospedeiro, através de pastas compartilhadas. Com isso, todos os dispositivos removíveis dos clientes ficam acessíveis através de um diretório de rede nos SOs virtualizados que é automaticamente disponibilizado pelo mecanismo de “Pastas

Compartilhadas" do *VirtualBox* (vide seção 3.7.6). O responsável por esta mediação entre o SSHFS e o *VirtualBox* é o Módulo de Controle de Privilégios.

Toda troca de dados entre o Módulo de Comunicação Servidor e os clientes é realizada através de serviços do SSH: SCP, no caso de informações provenientes da aplicação *FreeNX Server*, e SFTP, no caso de dados transferidos pelo SSHFS. A Figura 20 ilustra este cenário. Como o projeto WSE-OS utiliza rede sem fio, onde a transmissão está sujeita à interceptação e perda de dados, o Módulo de Comunicação Servidor implementa o algoritmo 3-DES (*Triple Data Encryption Standard*) em modo CBC (*Cipher Block Chaining*) para criptografia dos dados enviados e algoritmo MAC *hmac-sha1* para computação de integridade dos dados. A autenticação de usuário no ambiente WSE-OS é realizada através de usuário e senha cadastrada no servidor WSE-OS. No entanto, o cliente realiza, anteriormente, uma autenticação de servidor para a prevenção do ataque *man in the middle* (MEYER & WETZEL, 2009). Para isto, o sistema trabalha com chaves públicas e algoritmo de compartilhamento da *Pre-Shared Key* (PSK) *diffie-hellman-group1-sha1* [RFC2409] com chaves *ssh-rsa* de 512 bits e certificados *pgp-sign-rsa* para a autenticação explícita do servidor WSE-OS.

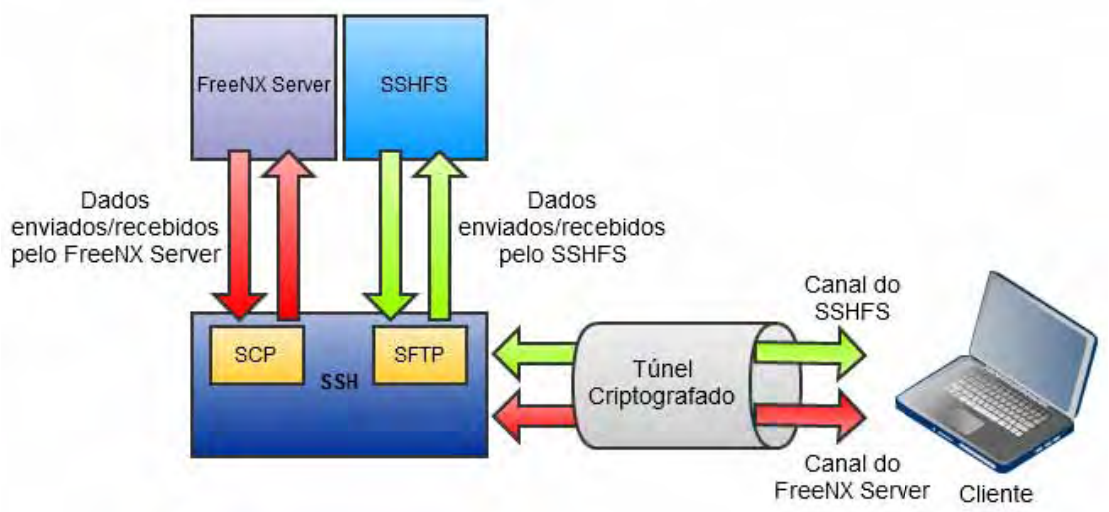

Figura 20 - Funcionamento do SSH no Módulo de Comunicação Servidor.

5.4 Mecanismo de Virtualização WSE-OS

O Mecanismo de Virtualização WSE-OS consiste em uma camada de *software* localizada no servidor WES-OS sendo responsável por colocar em execução os sistemas operacionais

solicitados pelos clientes. É este módulo que permite que um único computador (o servidor) seja capaz de executar concorrentemente diversos SOs, mantendo o isolamento de cada instanciação de forma a impedir que o funcionamento de um SO possa interferir no de outro.

Para o módulo de Mecanismo de Virtualização as seguintes exigências devem ser atendidas:

- Ser uma ferramenta gratuita, possuindo licença GPL, LGPL, FreeBDS ou PUEL para fins de pesquisa;
- Oferecer suporte a sistemas convidados 32-bit;
- Disponibilidade de versão para execução sobre um SO hospedeiro *Linux*;
- Mecanismo de virtualização que não exija uma versão modificada do SO convidado;
- Virtualização compatível com processadores que não possuam extensões de virtualização;
- Ter suporte à virtualização sobre a arquitetura x86.

Além de tais exigências, algumas características seriam desejáveis para a ferramenta de virtualização a ser utilizada na Camada de Gerenciamento WSE-OS:

- Ter suporte à virtualização sobre a arquitetura x64;
- Possibilitar a utilização de extensões de virtualização para computadores dotados de processadores com tecnologias VT-x e AMD-V;
- Oferecer suporte a sistemas convidados 64-bit.

A arquitetura da Camada de Gerenciamento WSE-OS possibilita que qualquer ferramenta de virtualização que atenda aos requisitos anteriormente listados seja utilizada no Mecanismo de Virtualização, porém para o modelo aqui proposto é empregado o uso da ferramenta *VirtualBox* (o Anexo A justificativa esta escolha e apresenta uma tabela comparativa entre as ferramentas de virtualização).

Para oferecer as funcionalidades necessárias para o projeto atendendo às exigências e características desejáveis, o módulo Mecanismo de Virtualização WSE-OS utiliza a ferramenta *VirtualBox* na versão 4.0, uma solução gratuita, que realiza a virtualização completa sem exigir que o processador possua extensões de virtualização (vale lembrar que o *VirtualBox* também permite utilizar tais extensões) e que apresentou um excelente

desempenho nos testes realizados na seção 6.2. Além disso, tal solução também apresenta versões para execução sobre várias plataformas operacionais (inclusive o *Linux*, que é utilizado pelo WSE-OS), tanto x86 quanto x64, e permite a execução de grande lista de sistemas operacionais convidados, inclusive sistemas 64-bit.

Em sua configuração padrão, o *VirtualBox* apresenta um aspecto limitante para o modelo WSE-OS: a forma como executa suas MVs a partir do disco rígido virtual. Para cada disco rígido virtual, o VMM do *VirtualBox* associa um identificador universal único (UUID). Desta forma, fica impossibilitada a instanciação de diversas execuções concorrentes a partir de um único disco rígido virtual (ou ISU), uma característica indesejável no contexto da solução WSE-OS. Para contornar este problema, é possível modificar a estrutura do descritor do arquivo que representa o disco rígido virtual do *VirtualBox*, para que ele seja marcado como *immutable* (CRUZ, 2008), fazendo com que diversas MVs possam utilizar concorrentemente a mesma ISU. Utilizando esta solução, o disco virtual não poderá mais ser utilizado em modo de leitura e escrita, ficando com o segundo modo de uma maneira temporária, ou seja, todas as alterações realizadas nos SOs pelos usuários, como configurações do sistema e arquivos gravados em disco, serão perdidas no desligamento da MV. Do ponto de vista de gerenciabilidade, esta é uma característica desejável, pois se mantém as ISUs sempre inalteradas. Entretanto, para os administradores de ambientes, este é um recurso limitante, pois impede futuras alterações nos SOs instalados nas ISUs.

Como forma de contornar este segundo impasse, pode-se desenvolver uma solução elaborada para permitir que um disco rígido virtual possa ser utilizado em modo *immutable* e *normal* ao se combinar duas cópias do arquivo com o disco rígido virtualizado. Neste caso, sempre haverá duas cópias de cada ISU: uma no modo *immutable* para permitir que várias máquinas virtuais possam utilizá-la concorrentemente e uma no modo *normal*, permitindo que o administrador do ambiente possa fazer alterações necessárias, e posteriormente gerar uma versão no modo imutável para substituir a previamente utilizada.

Para adequar o *VirtualBox* à solução WSE-OS, deixando a execução dos sistemas operacionais utilizados pelos usuários o mais próximo possível da execução nativa, foram realizadas três personalizações na solução de virtualização:

- Tratamento das teclas de atalho para serviços do *VirtualBox*: caso o usuário pressionasse qualquer combinação válida das teclas “HOST” + “Função”, por exemplo, “CTRL + F”, ativava-se uma função do VMM, neste caso, a MV sairia do modo tela-cheia, voltando para o modo janela. Com o tratamento citado, foram retiradas todas as teclas de atalho do *VirtualBox*. Desta forma, os usuários não conseguem ativar funcionalidades através de tais atalhos, o que seria uma característica indesejável para o ambiente WSE-OW;
- Remoção da Barra de Menus: por padrão, o *VirtualBox* apresenta um *menu* para ativação de determinadas funcionalidades da MV. Na solução WSE-OS, esta barra foi retirada para impedir que os usuários ativem tais funcionalidades;
- Remoção da Barra de Status: o *VirtualBox* apresenta, na porção inferior da janela, uma barra contendo diversas informações sobre a MV em execução. Na solução WSE-OS, esta barra foi retirada por questões visuais, para que o sistema aparente estar sendo executado diretamente sobre uma máquina física.

5.5 Módulo de Controle de Privilégios

O módulo de Controle de Privilégios consiste em um *script,* implementado em Python (LUTZ, 2001) (versão 2.6.4) em conjunto do *toolkit* GTK+ 2 (THE GTK+ TEAM, 2010) (versão 2.20.1), que entra em execução a cada nova solicitação de conexão recebida pelo servidor e é responsável por realizar o controle de acesso aos diversos usuários do WSE-OS. Além disso, este *script* faz o papel de mediador entre os módulos de Comunicação Servidor e Mecanismo de Virtualização com o SO hospedeiro do servidor, ou seja, é este *script* que coordena a interação entre tais módulos, de acordo com os privilégios de cada usuário que está ingressando no ambiente.

A interface do WSE-OS com o usuário foi desenvolvida para ser simples e intuitiva, permitindo que usuários ingressem no ambiente com facilidade e segurança. Em conjunto com o *FreeNX Server*, ela permite que o cliente tenha acesso somente à tela de escolha de sistema operacional, na qual serão listados apenas os SOs aos quais cada cliente tem acesso, não deixando possibilidade de que usuários mal intencionados tenham contato direto com o

sistema operacional hospedeiro do servidor WSE-OS. A Figura 21 apresenta um exemplo da interface.

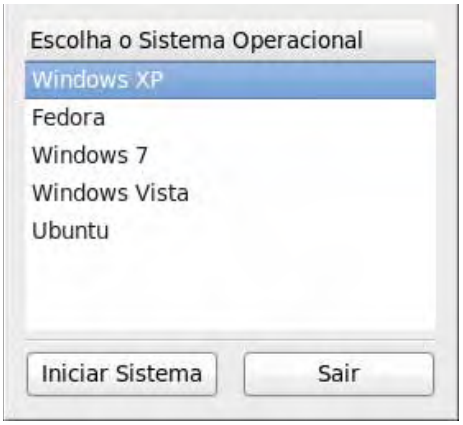

Figura 21 - Interface gráfica para escolha de sistema operacional.

A janela disponibilizada para escolha do sistema operacional a ser iniciado não apresenta opções de maximizar, minimizar e fechar para evitar que usuários acionem funcionalidades indesejáveis para o ambiente WSE-OS. Com isso, a janela permite apenas a escolha de um SO, sua inicialização e a finalização do acesso, reduzindo as possibilidades a apenas funcionalidades necessárias para o funcionamento da solução WSE-OS.

Dentre as funcionalidades do Módulo de Controle de Privilégios está a de realizar todos os passos necessários para configurar o SSHFS a fim de permitir que os usuários possuam acesso a seus dispositivos removíveis. A cada cliente que ingressa no ambiente, é iniciado um procedimento para montagem, no servidor WSE-OS, dos diretórios remotos localizados nas estações de trabalho. Outra funcionalidade também de responsabilidade do Módulo de Controle de Privilégios é a de colocar em execução o *VirtualBox* com as imagens de SOs escolhidas pelos usuários.

5.6 Base Operacional

A escolha da base operacional do servidor (SO hospedeiro) é norteada pela gratuidade, eficiência, segurança, estabilidade, pelo seu suporte as variações de periféricos e pela sua facilidade de instalação. Desta forma, é utilizado o sistema operacional *Fedora 13* para

arquiteturas x64, por possuir estabilidade e segurança reconhecida e por dar suporte à grande parte dos periféricos existentes no mercado.

O *Fedora* é uma versão baseada no *Red Hat*[39], de distribuição livre, que contém apenas *software* livre, de grande aceitação no mundo *Linux*, e tem ciclo rápido de desenvolvimento, com lançamento de novas versões a cada 6 meses em média. Além disso, possui uma interface amigável e grande facilidade de instalação/remoção de aplicativos através do *yum* (*Yellow dog Updater*, *Modified*), um *software* desenvolvido pela *Duke University* para ser um instalador, atualizador e removedor de pacotes RPM, similar ao *apt-get* do *Debian*. Vale ressaltar que a Camada de Gerenciamento WSE-OS também é aplicável a outras distribuições *Linux*, com possíveis modificações nas configurações de SSH, *FreeNX Server*, *VirtualBox* e SSHFS, dependendo das particularidades de funcionamento destas ferramentas sobre as diversas distribuições.

É importante observar que para se obter um maior desempenho, foram instalados somente os serviços necessários para o funcionamento do sistema operacional e para configuração do computador como servidor WSE-OS, a fim de operar com o mínimo de recursos necessários possível, melhorando assim a sua execução.

5.7 Funcionamento da Camada de Gerenciamento WSE-OS

Para um melhor entendimento do funcionamento da Camada de Gerenciamento WSE-OS, sendo também possível a verificação dos pontos de interação entre os módulos integrantes, é apresentado, pela Figura 22, um fluxo com as principais etapas do ciclo de execução dessa camada.

[39] http://www.br.redhat.com/

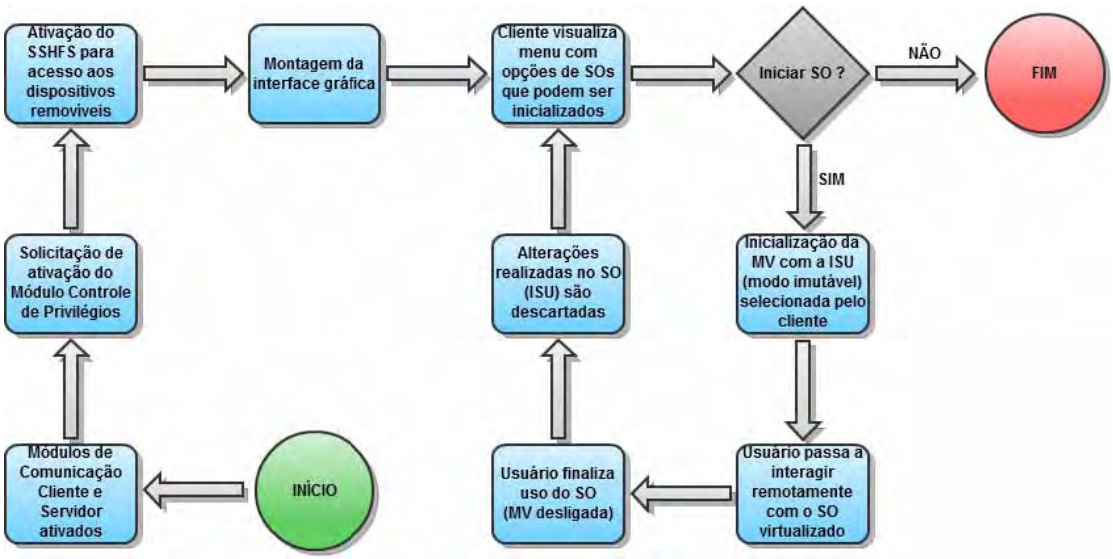

Figura 22 - Fluxo de execução da Camada de Gerenciamento WSE-OS

Através da análise da Figura 22 verifica-se que o fluxo tem início a partir do estabelecimento de conexão entre cliente e servidor. Posteriormente, o *middleware* envia uma solicitação de ativação do Módulo de Controle de Privilégios que, por sua vez, configura o SSHFS para permitir que o sistema operacional hospedeiro do servidor tenha acesso aos dispositivos removíveis do usuário. Em um segundo momento, o *script* de controle de acesso gera uma interface gráfica que relaciona apenas os SOs aos quais o usuário tem acesso, podendo este escolher qual SO deve ser instanciado. Com a escolha, o mecanismo de virtualização coloca em execução uma MV com o SO selecionado pelo cliente, que poderá interagir remotamente com esse sistema enquanto for de seu desejo. Assim que é finalizado o uso do SO pelo usuário, a MV é desligada, sendo descartadas todas as alterações realizadas no disco rígido virtual e novamente é apresentada a interface gráfica. A conexão entre cliente e servidor apenas é finalizada quando o usuário escolhe a opção “Sair” da janela de escolha apresentada pela Figura 21.

O comportamento de execução de um sistema operacional sobre a Camada de Gerenciamento WSE-OS é fiel ao funcionamento de um sistema operacional executado nativamente em um computador comum a partir do disco rígido local. Durante os testes, foi utilizada uma ISU com a versão do sistema operacional *Windows XP Service Pack 3* e suíte de aplicativos para escritório *Microsoft Office 2007*. O comportamento do sistema virtualizado e

todas as suas funcionalidades e recursos mantiveram-se idênticas ao da execução nativa do mesmo em um computador normal, com exceção dos seguintes pontos:

- No início do processo de *boot* do sistema operacional convidado, é apresentada uma tela de inicialização do *VirtualBox*;
- As configurações de vídeo do sistema convidado estão limitados aos recursos de vídeo emulados pelo *VirtualBox*. Atualmente, é possível utilizar os recursos da placa aceleradora de vídeo para aceleração 2D, o que torna a velocidade e as cores da reprodução de vídeos idênticas à original. Já a aceleração 3D, ainda é uma funcionalidade experimental no *VirtualBox*. O *driver* emulado configura a resolução do sistema virtualizado com a mesma resolução utilizada no *middleware* WSE-OS. Como neste a resolução é automaticamente definida em tempo de *boot*, conforme cada cliente, a interface do ambiente WSE-OS é sempre ajustado de acordo com as configurações de *hardware* de cada estação de trabalho;
- A transmissão de áudio para os clientes não é realizada pela Camada de Gerenciamento, pois como o modelos WSE-OS é especificamente projetado para trabalhar sobre redes sem fio, onde as taxas de transmissão são um limitante, optou-se por preservar um bom desempenho do ambiente, evitando atrasos na comunicação;
- Pelo fato de os dispositivos de armazenamento removíveis estarem localizados remotamente nos clientes, ainda não foi desenvolvido um mecanismo de leitura de dispositivos suficientemente transparente como em um computador normal. Atualmente, tais dispositivo são montados no ambiente de execução da Camada de Gerenciamento (*Fedora 13*), através do SSHFS, para então serem exportados para o *VirtualBox*, através da funcionalidade “Pastas Compartilhadas” do mesmo;
- Como as ISUs são iniciadas em modo *immutable* (protegido), ou seja, não é possível que o usuário altere de forma permanente o estado do disco rígido virtual, todos os dados e configurações do ambiente salvas pelos usuários são perdidos no próximo *boot*. Apesar de esta característica ser limitante, ela foi estabelecida como um requisito do ambiente, para impedir que usuários possam degradar as ISUs de sistemas, que podem ser compartilhadas por diversos usuários.

Apesar das limitações acima listadas, estas também são encontradas em todas as outras soluções de virtualização e ambiente de trabalho remoto, licenciadas ou de código fonte aberto. Além disso, o funcionamento da Camada de Gerenciamento garante que qualquer sistema operacional suportado pelo *VirtualBox*, com seus possíveis aplicativos, seja executado sem a necessidade de alteração em seu código fonte ou modificações na forma de instalação, tendo funcionamento igual ao de um SO com execução nativa.

5.8 Considerações Finais

Este capítulo apresentou a arquitetura da Camada de Gerenciamento WSE-OS, analisando seus módulos, características e funcionamento. Através desta análise é possível observar que a Camada de Gerenciamento WSE-OS integra três ferramentas fundamentais independentes funcionalmente entre si, tirando proveito desta combinação a fim de oferecer uma solução de gerenciamento centralizado de computadores através de ISUs, apresentando as seguintes características:

- Acesso, por parte dos clientes, a *desktops* virtuais através de redes sem fio;
- Troca de dados entre clientes e servidor de maneira segura;
- Abstração de *hardware* oferecida pelo mecanismo de virtualização, permitindo que diversos sistemas operacionais possam estar em funcionamento concorrentemente sobre o mesmo computador e de maneira isolada;
- Acesso, por parte do servidor, aos dispositivos removíveis dos clientes WSE-OS através de um mecanismo que provê a implementação de um sistema seguro de arquivos remotos em espaço de usuário.

Através da análise das características apresentadas nesta seção é possível perceber que a configuração da Camada de Gerenciamento WSE-OS é adaptada para ambientes com redes sem fio, por utilizar soluções que prezam por taxas de transmissão reduzidas e confidencialidade e integridade dos dados transmitidos, permitindo que diversos clientes se utilizem de sistemas remotos de maneira rápida e segura. Além disso, tal configuração se preocupa com a estabilidade do ambiente, executando de maneira isolada cada sistema operacional utilizados pelos usuários.

6. EXPERIMENTOS

6.1 Considerações Iniciais

Este Capítulo apresenta os resultados dos experimentos desenvolvidos neste trabalho e que consistiram em extrair uma base de informações de desempenho da Camada de Gerenciamento WSE-OS. De início, foi realizada uma análise comparativa entre o *VMware Workstation*, *VirtualBox* e o QEMU, para embasar a escolha feita por este trabalho. Por fim, são apresentados testes aplicados na Camada de Gerenciamento WSE-OS a fim de observar o tempo de inicialização dos sistemas operacionais e o número de usuários concorrentes suportados pelo sistema. Com os resultados obtidos dos experimentos foi possível verificar a viabilidade do modelo de virtualização centralizada proposto pelo WSE-OS, sua capacidade de escala e a degradação introduzida pelas técnicas utilizadas na solução.

6.2 Análise entre as Ferramentas de Virtualização

Nesta seção, são apresentados os resultados da análise comparativa entre o *VMware Workstation*, *VirtualBox* e o QEMU (em conjunto com seu acelerador KQEMU). Para tal comparação, foi utilizado o *software Phoronix Test Suite* em sua versão 2.8.2 (PHORONIX MEDIA, 2011), um conjunto de ferramentas para realização de *benchmarking*.

O *Phoronix Test Suite* é a mais abrangente plataforma de testes e *benchmarking* disponível para o sistema operacional *Linux*. Esta ferramenta permite realizar de maneira fácil e eficaz comparações tanto qualitativas como quantitativas. É baseado em vários trabalhos de *benchmarking* para *Linux* e ferramentas internas desenvolvidas pela Phoronix.com desde o ano de 2004, juntamente com a parceria de vinte e um desenvolvedores de *hardware* e *software* para computadores. É uma ferramenta *open source*, licenciada sob a GPL do GNU (PHORONIX MEDIA, 2010).

O *Phoronix Test Suite* possibilita a realização de diversos tipos de testes, que podem ser vistos na documentação[40] do mesmo. Para a análise realizada neste trabalho, foram escolhidos os seguintes testes:

- *IOzone*: testa a velocidade do disco para leitura e escrita, permitindo a escolha do tamanho do arquivo. Neste trabalho, foi utilizado um arquivo de 512 MB de tamanho. O teste tem como unidade de medida os MB/s (*Megabytes* por segundo);
- *LZMA-compress*: testa o tempo para compressão de um arquivo de 256 MB de tamanho, utilizando o algoritmo de compressão de dados de *Lempel-Ziv-Markov*[41]. O teste tem como unidade de medida os segundos (s);
- *7zip-compress*: testa a velocidade de compressão que o sistema atinge utilizando a ferramenta *7-Zip*. A unidade de medida utilizada por este teste é a MIPS (Milhões de Instruções por Segundo);
- *SQLite*: teste que mede o tempo para executar um número pré-determinado de inclusões em um banco de dados indexado. No caso deste trabalho foram utilizadas 3000 inclusões. O teste tem como unidade de medida os segundos;
- *BYTE Unix Bench*: consiste em um conjunto de testes individuais que são direcionados a áreas específicas. Neste trabalho será o utilizado o modelo *Dhrystone*[42] desenvolvido em 1984 por Reinhold P. Weicker. Tal modelo é normalmente utilizado para medir e comparar o desempenho de computadores. O *Dhrystone* não executa operações de ponto flutuante, mas envolvem matrizes, caracteres de *string*, endereçamento indireto, e a maioria das instruções (não envolvendo ponto flutuante) que podem ser encontrados em um aplicativo (PHORONIX MEDIA, 2009). O resultado deste teste é número de *dhrystones* por segundo (o número de iterações do *loop* do código principal por segundo).

Os testes foram realizados em um computador com as seguintes especificações:

- Processador Intel *Core* 2 *Duo* E4500 2.20 GHz;
- Memória DDR RAM de 3 GB;
- Placa de vídeo nVIdea GeForce 7300 GS com 256 MB DDR2;
- Disco rígido SATA com 300 GB de capacidade e 7200 RPM;

[40] http://www.phoronix-test-suite.com/?k=documentation.
[41] http://7zip.rnbastos.com/sdk.html.
[42] http://www.inf.ufrgs.br/gppd/disc/cmp134/trabs/T1/001/benchmarks/Benchmarks-dhrystone.htm.

- Partição *swap*[43] de 4GB.

O sistema operacional escolhido para ser o sistema hospedeiro do computador descrito é o *Fedora 13* (de codinome *Goddard*), lançado em junho de 2010. A versão está dotada de *Kernel Linux* 2.6.33.3-85.fc13.i686.PAE, gerenciador de janelas GNOME 2.30.0 e sistema de arquivos Ext4.

Para a realização dos testes, foram criadas máquinas virtuais para cada ferramenta de virtualização. Todas as MVs foram configuradas com sistema operacional *Ubuntu 8.04 LTS* "*Hardy Heron*" com 1024MB de RAM e um disco virtual de 10 GB. O SO convidado possuía *Kernel* 2.6.24-27-*generic* (i686), GNOME 2.22.3, servidor *X.Org* 1.4.0.90 e sistema de arquivos Ext3. As máquinas virtuais foram criadas utilizando o arquivo de disco padrão de cada solução, ou seja, *VirtualBox* com .vdi, o *VMware Workstation* com .vmdk e o QEMU com o .qcow2.

As subseções seguintes apresentam os resultados da análise comparativa entre as ferramentas de virtualização *VMware Workstation*, *VirtualBox* e QEMU. Todos os valores apresentados representam a média calculada a partir de dez execuções de cada teste. Também são calculados para cada experimento os desvios padrões, possibilitando a visualização das dispersões estatísticas, e os intervalos de confiança em nível de 90%, considerando a distribuição *t de Student* (FARHAT, 2008).

6.2.1 Teste com o IOzone

Ambos os testes realizados com o *IOzone*, consideraram um arquivo de 512 MB de tamanho. O teste de velocidade de escrita em disco apresentou um cenário de equivalência entre *VMware Workstation* (com 65,51 MB/s) e *VirtualBox* (com 61,53 MB/s) e disparidade do QEMU (com 16,49 MB/s), como pode ser visto na Figura 23. Os desvios padrões para os valores da Figura 23 são de 1,22, 1,32 e 2,14 MB/s para o *VMware*, *Virtualbox* e QEMU, respectivamente. Os intervalos de confiança são [64,80, 66,22], [60,76, 62,30] e [15,25, 17,73] para o *VMware*, *Virtualbox* e QEMU, respectivamente.

[43] Parte do disco que é usada como memória virtual.

No teste de leitura do disco, o *VirtualBox* conseguiu superar o *VMWare* em aproximadamente 10%. Já o QEMU, ficou numa desvantagem ainda maior em relação aos concorrentes (51% mais lento em relação à média de desempenho entre *VMware Workstation* e *Virtualbox*), conforme a Figura 24. Os desvios padrões para os valores da Figura 24 são de 5,46, 4,86 e 5,93 MB/s para *VMware*, *Virtualbox* e QEMU, respectivamente. Os intervalos de confiança são [1839,30, 1845,62], [2009,72, 2015,36] e [986,73, 993,61] para o *VMware*, *Virtualbox* e QEMU, respectivamente.

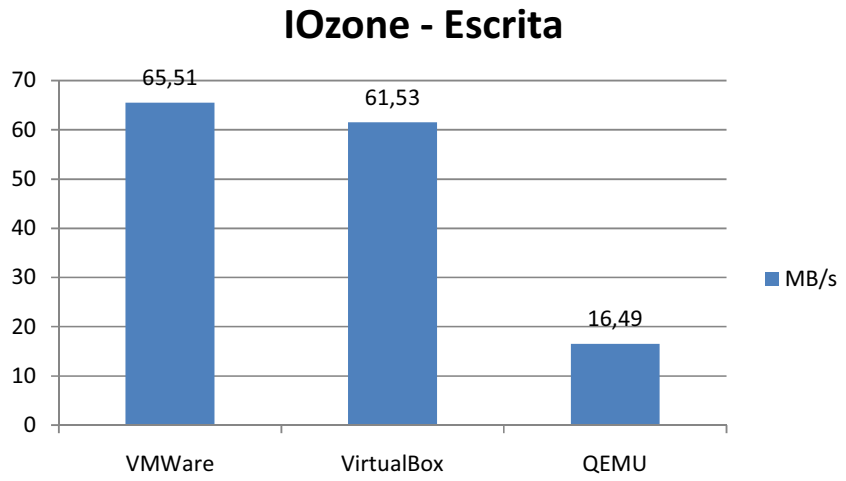

Figura 23 - Teste de velocidade de escrita em disco com o *IOzone*.

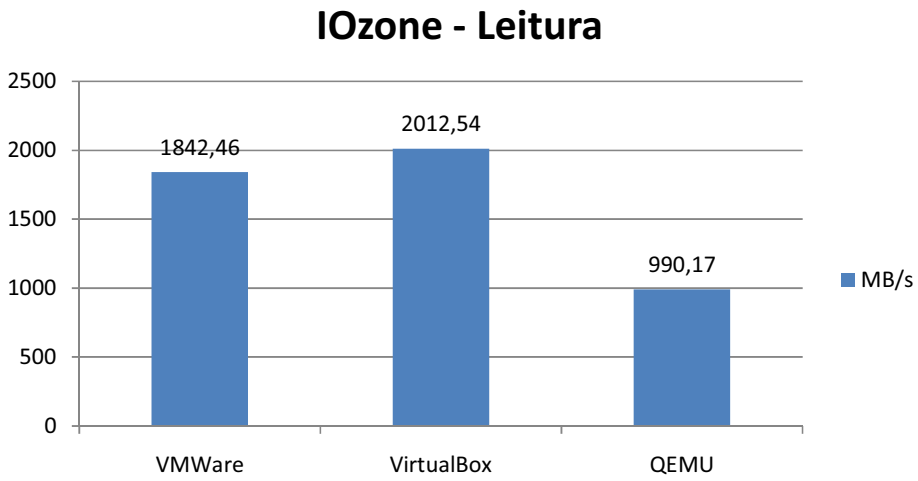

Figura 24 - Teste de velocidade de leitura de disco com o *IOzone*.

6.2.2 Teste com o LZMA-Compress

O resultado obtidos com o teste LZMA apresentou um desempenho muito próximo entre o *VMware Workstation*, com 337,97 segundos, e o *VirtualBox*, com 338,96 segundos. Em contraste a estes resultados, o QEMU obteve uma desvantagem de aproximadamente 10 vezes mais lento (comparação realizada com a média dos tempos das soluções concorrentes). A Figura 25 expõe os resultados obtidos com desvios padrões de 2,76, 2,03 e 8,45 segundos para *VMware*, *Virtualbox* e QEMU, respectivamente. Os intervalos de confiança são [336,37, 339,57], [337,78, 340,14] e [3372,71, 3382,51] para o *VMware*, *Virtualbox* e QEMU, respectivamente.

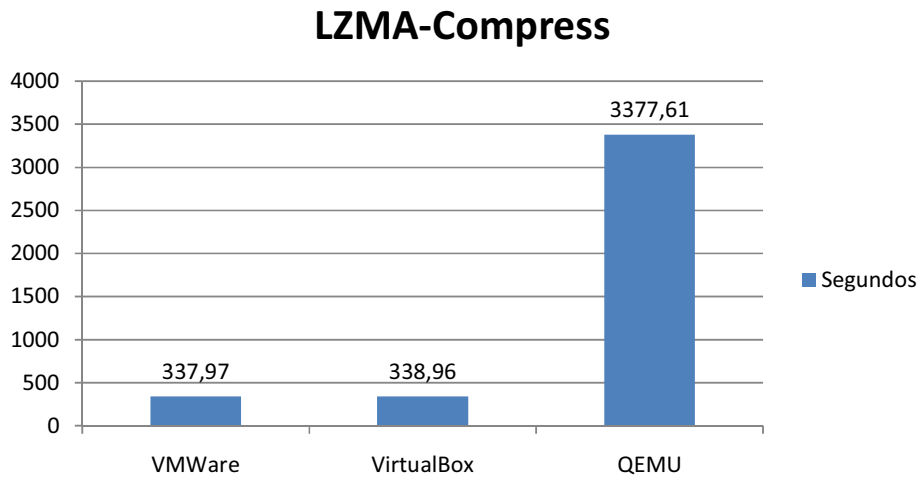

Figura 25 - Teste com *LZMA compress*.

6.2.3 Teste com o 7zip-*Compress*

O teste realizado com o *7zip-Compress Benchmark* mediu a velocidade de compactação em cada ambiente. Observa-se na Figura 26 que a ferramenta *VMware Workstation* apresentou o melhor desempenho, seguido de perto pela *VirtualBox*. Já a solução QEMU apresentou números bem inferiores. Os desvios padrões para os valores da Figura 26 são de 7,21, 6,85 e 7,65 MIPS para *VMware*, *Virtualbox* e QEMU, respectivamente. Os intervalos de confiança são [1604,82, 1613,18], [1500,03, 1507,97] e [415,57, 424,43] para o *VMware*, *Virtualbox* e QEMU, respectivamente.

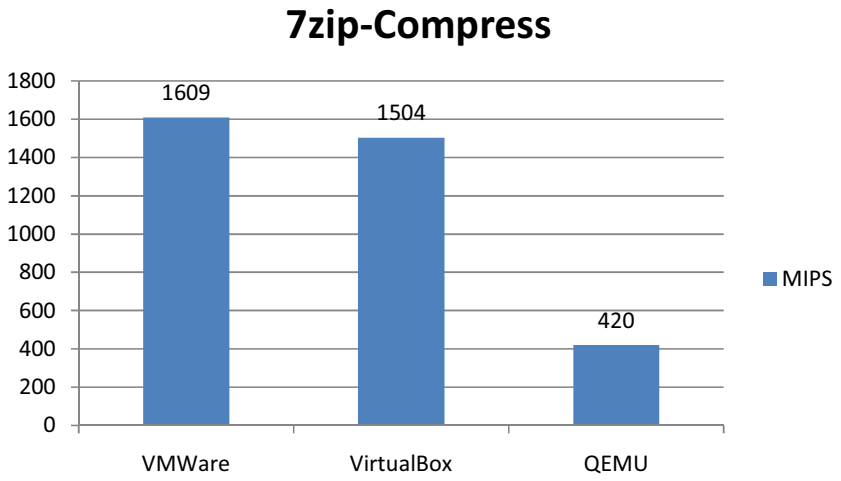

Figura 26 - Teste com *7-zip compress*.

6.2.4 Teste com o *SQLite*

O teste com o *SQLite* considerou um montante de 3000 inserções em um banco de dados. O resultado foi o que apresentou maior disparidade entre *VMware Workstation* e *VirtualBox*, ficando o segundo cerca de 2,84 vezes mais lento em relação ao primeiro, conforme a Figura 27. O QEMU ficou 8,40 vezes mais lento que o *VMware Workstation*. Os desvios padrões para os valores da Figura 27 são de 0,98, 1,12 e 1,74 segundos para *VMware*, *Virtualbox* e QEMU, respectivamente. Os intervalos de confiança são [17,67, 18,81], [51,16, 52,46] e [152,38, 154,40] para o *VMware*, *Virtualbox* e QEMU, respectivamente.

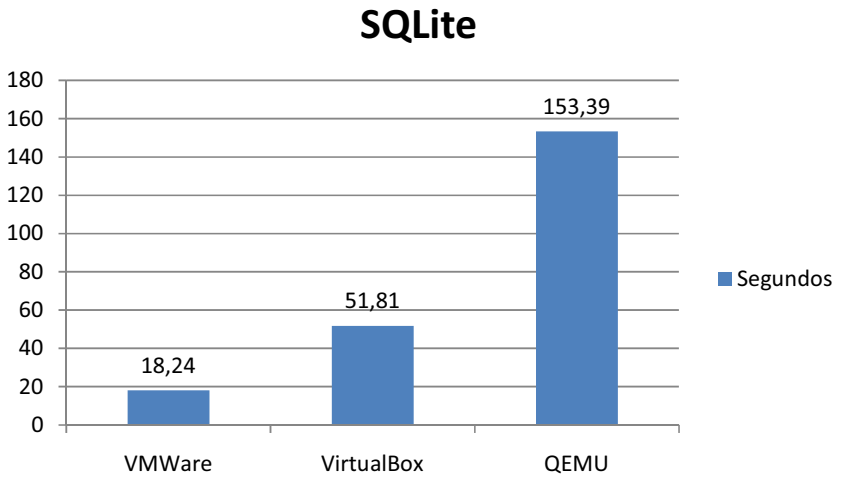

Figura 27 - Teste com *SQLite*.

6.2.5 Teste com o *BYTE Unix Bench*

No teste realizado com o *BYTE Unix Bench*, o *VirtualBox* apresentou o melhor desempenho, com 7211057,1 LPS, seguido de perto pelo *VMware Workstation*, com 7336292,4 LPS. Não foi possível realizar o teste para o QEMU, pois a máquina virtual travou em todas as tentativas de aplicação do *benchmark*. A Figura 28 apresenta os resultados obtidos com desvios padrões de 20,32 e 23,73 LPS para *VMware* e *Virtualbox*, respectivamente. Os intervalos de confiança são [7211045,32, 7211068,88] e [7336278,65, 7336306,16] para *VMware* e *Virtualbox*, respectivamente.

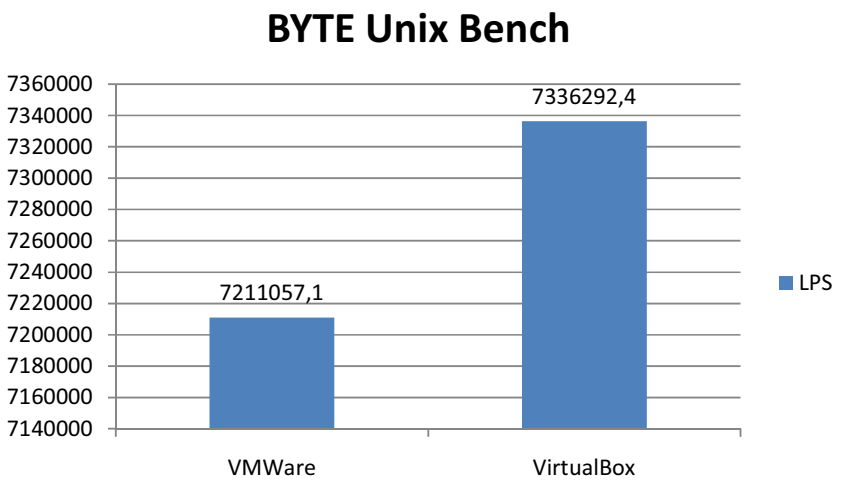

Figura 28 - Teste com *BYTE Unix Bench*.

6.3 Desempenho da Camada de Gerenciamento

A comunicação TCSC e módulo de virtualização associados ao processamento centralizado impõem uma compensação negativa pelos benefícios da flexibilidade, abstração de *hardware*, da possibilidade de execuções concorrentes de sistemas operacionais em um único computador e do compartilhamento de recursos. Tal compensação consiste em um decréscimo no desempenho do sistema.

Conforme apresentado no Capítulo 5, para atender aos requisitos de funcionamento impostos pela Camada de Gerenciamento WSE-OS, foi considerado, para fins de testes, a

ferramenta de virtualização *VirtualBox* (solução escolhida para o modelo WSE-OS). O objetivo destes experimentos é avaliar o impacto que a comunicação TCSC, o módulo de virtualização e a centralização do processamento em um servidor causam no desempenho da solução WSE-OS.

Os testes consistiram em realizar instanciações simultâneas de máquinas virtuais utilizando a mesma imagem de sistema operacional sobre um servidor e analisar o tempo de *boot* destes sistemas. Estes testes foram realizados tanto localmente (a partir do próprio servidor), quanto remotamente, através de estações remotas. Para isto, foram utilizados clientes dotados do *Middleware* WSE-OS para permitir o acesso ao servidor. Todos os valores apresentados representam a média calculada a partir de dez execuções de cada teste. Também são calculados para cada experimento os desvios padrões, possibilitando a visualização das dispersões estatísticas, e os intervalos de confiança em nível de 90%, considerando a distribuição *t de Student*. O ambiente utilizado nos experimentos é apresentado na Tabela 2:

Tabela 2 - Configuração dos computadores e roteador do ambiente de testes.

Tipo	Configuração
Servidor	Intel Xeon E5540 @2.53GHz Gainestown 45nm T, MB Intel S5520SC, 16GB Triple-Channel DDR3 @ 533MHz, HD 1TB Western WDC ATA. Sistema operacional hospedeiro Fedora 13 dotado de Kernel Linux 2.6.33.5-112.fc13.x68_64, gerenciador de janelas GNOME 2.30.0, sistema de arquivos Ext4, memória SWAP de 16GB e VirtualBox V4.0.
Cliente 1	Intel Pentium 4 HT @1GHz, MB ASUS P4P800 865pe, 640MB DDR @398MHz, Atheros Wireless Card AGN.
Cliente 2	Intel Dual Core E4500 @2.2GHz, MB Cent. IntelQ43, 1GB DDR @400MHz, Atheros Wireless Card AGN.
Cliente 3	Intel Core 2 Duo P8600 @2.4GHz, MB Sony Corp. VAIO GM47, 4GB DDR2 @398MHz, Intel WiFi Link 5100 AGN.
Cliente 4	Intel Core 2 Quad Q9400 @2.66GHz, MB Cent. IntelQ43, 4GB DDR3 @532MHz, Atheros Wireless Card AGN.
Roteador	300M sem fio N Router, Modelo No. TL-WR941ND.

Para coleta das informações referentes ao tempo de *boot* dos sistemas virtualizados foram utilizadas as suítes de *benchmark* SANDRA (SISOFTWARE, 2010) e *InterMapper Flows* (DARTWARE, 2011).

A ferramenta *InterMapper Flows* é uma solução para monitoramento de redes sem fio (padrões 802.11 a/b/g/n) capaz de capturar pacotes de dados trocados entre cliente e

servidor oferecendo análises úteis para que fosse possível a realização dos testes desta seção. Este analisador suporta uma vasta gama de protocolos permitindo uma visualização detalhada de transmissão e recebimento de pacotes clientes, desta forma provendo recursos para medição do tempo de inicialização de sistemas operacionais replicados nos clientes dotados do *middleware* WSE-OS através de comunicação remota TCSC.

A solução SANDRA (*System Analyser Diagnostic and Reporting Assistant*) é uma suíte de ferramentas para *benchmark* desenvolvidas para análise de desempenho de diferentes tipos de *hardware* como, por exemplo, processadores, placas gráficas, sistemas de memória, multimídia, rede e *motherboard*. Este aplicativo de monitoramento, no caso deste projeto, é utilizado para geração de relatórios sobre medições de tempo de *boot* de sistemas operacionais nativos e virtualizados no servidor WSE-OS.

Através da coleta de dados a partir desses dois aplicativos de monitoramento é possível correlacionar informações de execução local e remota, servindo como base para discussão das diferenças de desempenho e da viabilidade de uso da Camada de Gerenciamento WSE-OS associada ao *Middleware* WSE-OS como solução para replicação de sistemas operacionais em ambientes sem fio.

Para a realização dos testes, foi criada uma ISU, utilizando o arquivo de disco padrão do aplicativo *VirtualBox* (.vdi), de 10 GB de tamanho contendo a instalação de um *Windows XP Service Pack 3*. Esta ISU serviu de base para a criação de todas as MVs utilizadas no experimento, sendo que todas elas foram configuradas com 192 MB de memória RAM e para utilizarem as extensões de virtualização proporcionadas pelo processador do servidor (Intel VT-x). Além disso, a distribuição dos dados aos clientes através da rede sem fio e do uso do ambiente gráfico remoto foi implementado com o auxílio de *caches* em memória RAM e memória *Flash* (*Flash Driver USB*) com alocação de 16 MB e 64 MB, respectivamente em cada *middleware* cliente.

A Tabela 3 apresenta uma comparação para o ambiente com apenas um cliente. O tempo de *boot* (imagem de referência) ativado pela comunicação remota é 3,85 segundos mais lento que esta imagem virtualizada com acesso local no servidor WSE-OS, que por sua vez apresenta um aumento de 1,28 segundos em relação ao mesmo SO sendo executado

nativamente sobre o servidor. No primeiro caso, o atraso é em decorrência do meio de comunicação e do protocolo TCSC. Já na segunda comparação, a diferença fica a cargo do mecanismo de virtualização. É importante observar que esses tempos não contabilizam o tempo de *boot* do *Middleware* WSE-OS, sendo os valores referentes apenas à inicialização do sistema operacional virtualizado. A medição do tempo de *boot* remoto se deu em um computador com configuração do Cliente 3 da Tabela 2.

Tabela 3 - Tempo de *boot* da imagem de referência para um computador cliente.

Sistema Operacional	Modo de Inicialização da Imagem	Memória RAM alocada pelo Servidor	Tempo de *boot* (segundos)	Desvio Padrão	Intervalo de Confiança (*Student* – 90%)
Windows XP Service Pack3	Nativo com Acesso Local	16GB	14,05	0,33	[13,85, 14,24]
Windows XP Service Pack3	Virtualizado com Acesso Local	192MB	15,33	0,42	[15,08, 15,57]
Windows XP Service Pack3	Virtualizado com Acesso Remoto	192MB	19,18	0,25	[19,04, 19,33]

Para analisar a capacidade de escala deste ambiente proposto e avaliar o ponto limiar de saturação de virtualização concorrente aplicada a uma mesma imagem de sistema operacional, foram utilizados diferentes cenários para a medição do tempo de inicialização da máquina virtual. A Tabela 4 apresenta os resultados obtidos com os testes, levando em consideração diferentes quantidades de máquinas virtuais (MVs) com solicitações locais e remotas (através da comunicação TCSC) para virtualização da imagem de referência. A medição dos tempos de *boot* remoto se deu em um computador com configuração do Cliente 3 da Tabela 2. Já para os tempos de *boot* local, a medição foi realizada no próprio servidor.

Observando-se o tempo de *boot* local de um único sistema operacional virtualizado, 15,33 segundos, com o de um sistema operacional nativo sobre o mesmo servidor, que é de 14,05 segundos, constata-se uma mínima degradação, aproximadamente 9%, em função do mecanismo de virtualização. Para até 10 MVs concorrentes, é possível observar uma diferença inferior a 9 segundos, em relação a uma única MV, o que representa uma degradação de 55%. Com um maior número de MVs iniciando a mesma ISU concorrentemente, o desempenho do sistema cai gradativamente, sendo esta queda em função do compartilhamento dos recursos oferecidos pelo servidor.

Tabela 4 - Tempo de *boot,* em segundos, para máquinas virtuais concorrentes.

	LOCAL			REMOTO		
Cenário	Tempo de *boot* (segundos)	Desvio Padrão	Intervalo de Confiança (*Student* – 90%)	Tempo de *boot* (segundos)	Desvio Padrão	Intervalo de Confiança (*Student* – 90%)
01 MV	15,33	0,21	[15,21, 15,45]	19,18	0,25	[19,04, 19,33]
05 MVs	20,24	0,25	[20,10, 20,39]	23,27	0,28	[23,11, 23,43]
10 MVs	23,77	0,27	[23,61, 23,93]	26,95	0,31	[26,77, 27,13]
15 MVs	27,13	0,31	[26,95, 27,31]	30,46	0,36	[30,25, 30,67]
20 MVs	34,07	0,35	[33,87, 34,27]	37,55	0,41	[37,31, 37,79]
25 MVs	52,28	0,41	[52,04, 52,52]	55,91	0,47	[55,64, 56,18]
30 MVs	163,07	0,50	[162,78, 163,36]	167,00	0,57	[166,67, 167,33]
35 MVs	197,29	1,10	[196,65, 197,93]	201,53	1,23	[200,82, 202,24]
40 MVs	353,28	1,32	[352,52, 354,05]	357,82	1,45	[356,98, 358,66]
45 MVs	1125,07	10,43	[1119,02, 1131,12]	1129,91	12,34	[1122,76, 1137,06]

O desempenho para a configuração de servidor utilizado no teste é aceitável para até quarenta MVs concorrentes. No cenário de quarenta e cinco MVs concorrentes utilizando a mesma ISU, a degradação ultrapassa qualquer padrão observado nos demais casos. Isto ocorre pelo fato da quantidade de memória RAM utilizada pelas máquinas virtuais ultrapassar a metade de quantidade de memória existente no servidor, algo que não é recomendado pela própria Oracle (ORACLE CORPORATION, 2011). Esta sugere que seja reservado pelo menos 50% da memória RAM do servidor para o sistema operacional hospedeiro. No cenário de quarenta e cinco clientes, estes utilizam 8640 MB, totalizando aproximadamente 53% da memória física existente. Com o limiar ultrapassado, o desempenho do ambiente se torna relativamente inviável para o uso corporativo e para estações de trabalhos que exigem rapidez na inicialização de sistemas.

Observa-se também um aumento no tempo de *boot* quando os acessos aos sistemas virtualizados são realizado remotamente. Esta degradação se dá em função de dois atrasos, um gerado pelo meio de comunicação e o outro pelo *overhead* decorrente da criptografia/descriptografia, verificação de integridade e compressão/descompressão utilizadas. O primeiro é variável em função do número de clientes, já que a rede fica congestionada quando o número de conexões concorrentes aumenta. Já o segundo, consiste em um atraso fixo para cada cliente, pois os processos de criptografia/descriptografia, verificação de integridade e compressão/descompressão estão sempre ativos para os

usuários e depende da configuração de *hardware* de cada um (máquinas com maior poder de processamento apresentarão menor atraso). É importante observar que, apesar de o sistema estar configurado para utilização de *cache*, os resultados obtidos na Tabela 4 não refletem melhoria em função deste mecanismo. Isto se deve ao fato da medição dos tempos ter sido feita na primeira inicialização do sistema operacional, onde os *caches* ainda não possuem informações de execuções anteriores.

Para melhor visualização da condição limite imposta pela memória RAM disponível no servidor, a Figura 29 traça um gráfico referente aos dados da Tabela 4, correlacionando a porcentagem de degradação do tempo de *boot* de SOs remotamente virtualizados em função da porcentagem de uso de memória RAM para instanciação daqueles SOs em Servidor WSE-OS. Através desta figura é possível observar que, conforme previsto por (ORACLE CORPORATION, 2011), uma quantidade de máquinas virtuais concorrentes que somem um valor acima de 50% da memória total disponível em servidor WSE-OS reflete em uma degradação no tempo de inicialização de SO que inviabiliza o uso do sistema.

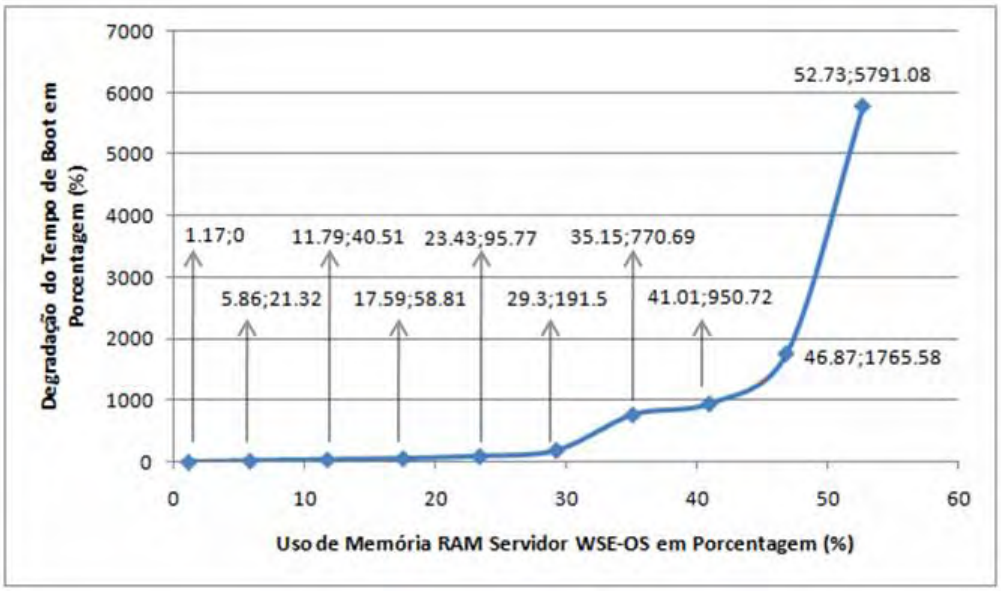

Figura 29 - Degradação no tempo de *boot* em função de memória utilizada em servidor de 16GB.

A Figura 29 confirma que, de fato, a quantidade de memória RAM disponível em servidor consiste no principal gargalo para a concorrência de clientes no ambiente WSE-OS. Em um cenário de replicação do ambiente com um servidor melhor dotado de RAM, o número de clientes conectados concorrentemente aumentaria desde que a quantidade de memória

alocada para cada máquina virtual (MV) fosse a mesma das MVs utilizadas nos testes deste capítulo (192MB). Apesar de a memória RAM do servidor ser o principal gargalo da solução WSE-OS, em cenários onde o número de clientes atinja um número elevado, outros parâmetros devem ser considerados. Supondo um ambiente com 1000 clientes, as capacidades de processamento do servidor e transmissão do ponto de acesso da rede também consistiriam em fatores limitantes a serem considerados.

Para verificar o desempenho do sistema WSE-OS em um servidor dotado de uma quantidade diferente de RAM, a Figura 30 apresenta a mesma análise realizada na Figura 29 sobre a mesma plataforma de *hardware* do teste anterior, porém, neste caso, com apenas 4GB de memória RAM em servidor. Nesta nova avaliação, a alocação de memória para as MVs também continua a mesma (192MB). Diante deste novo cenário, é possível observar que a escalabilidade do ambiente cai para dez conexões simultâneas, continuando o limite máximo de MVs concorrentes dependente dos 50% de alocação da memória RAM servidor para virtualização. É importante ressaltar que as porcentagens de degradações no tempo de boot da ISU de referência nos servidores de 16GB e 4GB mantêm um mesmo padrão, sendo visível a semelhança entre as curvas da Figura 29 e Figura 30.

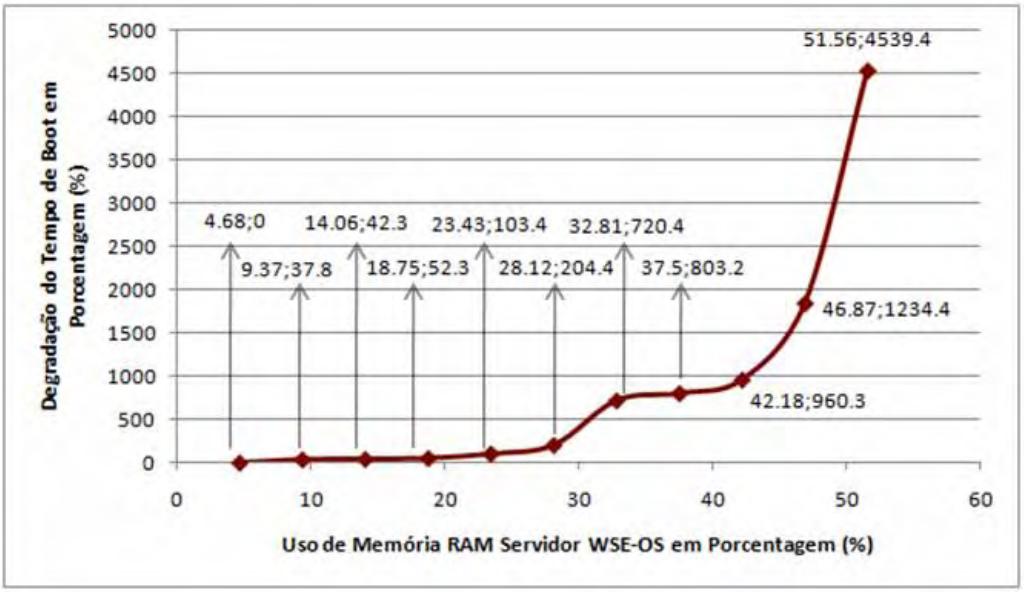

Figura 30 - Degradação no tempo de *boot* em função de memória utilizada em servidor de 4GB.

A Tabela 4 apresenta tempos de *boot* concorrentes de uma mesma imagem de SO como referência para análise da degradação causada pelo processamento de solicitações

simultâneas. Esta situação é uma das mais críticas possíveis em relação ao processamento, acesso a disco rígido e gerenciamento de memória do servidor e uma das mais onerosas para a rede de comunicação sem fio do ambiente, pois são nestes momentos que ocorrem o carregamento em memória do servidor de diversas imagens virtualizadas de SO, processamento do *boot* e distribuição de pacotes para os diversos clientes do ambiente.

A dinâmica de funcionamento do ambiente logo após a fase de carregamento em memória do servidor dos SOs virtualizados não demonstra atrasos significativos para as solicitações de aplicativos, mesmo que estas sejam simultâneas de várias estações clientes. É de se esperar que programas que exijam maior processamento como, por exemplo, ferramentas de edição de vídeos e imagens, apresentem maior latência de resposta, pois neste caso a capacidade de processamento do servidor é compartilhada por diversos usuários e consistiria em um gargalo para o sistema. Em um cenário mais próximo do real, onde os aplicativos possuem um consumo mínimo de processamento, pelo fato dos SOs já estarem em memória RAM do servidor, os desempenhos dos *software* em uso apresentam comportamentos próximo, ou até melhores, ao de uma execução nativa sobre *hardware* cliente, pois na maior parte do tempo o servidor fica com sua capacidade de processamento ociosa e, com isso, os clientes podem desfrutar de um nível mais elevado de desempenho, sendo este aumento em função de o *hardware* do servidor possuir melhor configuração em relação aos clientes.

Para uma visualização direta do nível de degradação causada pela virtualização e pela latência decorrente da comunicação remota, a Figura 31 compara o tempo de *boot* da imagem do sistema operacional *Windows XP Service Pack 3* em execução nativa sobre o Cliente 3 descrito na Tabela 2, com desvio padrão de 0,42 segundos e intervalo de confiança [47,60, 48,09], e em execução remota através do ambiente WSE-OS utilizando a mesma plataforma de *hardware* cliente, com desvio padrão de 0,37 segundos e intervalo de confiança [53.45, 53.88]. Esta figura também contabiliza o tempo de inicialização do *Middleware* WSE-OS e considera um cenário onde o *boot* do sistema operacional é realizado pela segunda vez, refletindo na redução da latência da comunicação TCSC em função do *cache* utilizado, possibilitando a visualização do ganho de desempenho em função deste mecanismo.

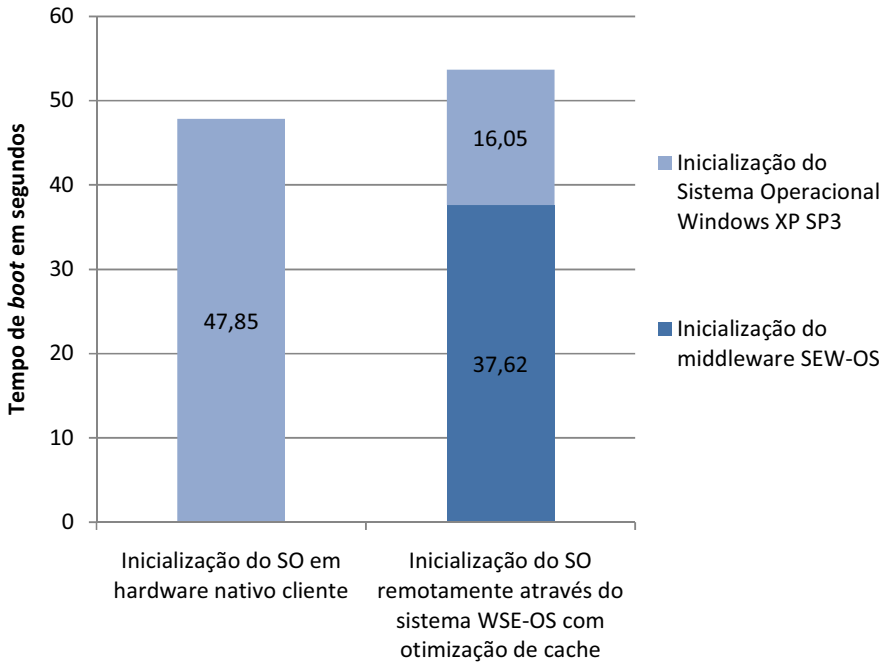

Figura 31 - Tempo de *boot* SO referência: execução nativa em cliente X execução sistema WSE-OS.

Neste teste de comparação, os resultados foram coletados na segunda instanciação do sistema operacional, onde os *caches* tanto do cliente quanto do servidor já estão alimentados (para melhor entendimento deste mecanismo, ver seção 2.4.4). A partir deste cenário, o sistema WSE-OS tem um acréscimo de 5,82 segundos comparado ao tempo de *boot* nativo em *hardware* cliente. Este pequeno aumento representa aproximadamente 12,16% de degradação para a instanciação de uma imagem de sistema operacional de referência pelas tecnologias utilizadas na solução WSE-OS associadas ao modelo de transmissão de dados em rede sem fio.

Através da Figura 31, pode-se observar uma redução de tempo de *boot* remoto do SO virtualizado (16,05 segundos), se comparado ao resultado apresentado pela Tabela 3 (19,18 segundos), sendo esta redução de 16,31% em decorrência do uso de *cache*. Ainda que o sistema esteja configurado para utilização de *caches* em memória RAM e memória *Flash Driver* USB nos dois cenários, este proveito só foi observado a partir da segunda instanciação da imagem de referência, pois neste segundo momento os *caches* já estavam alimentados com dados decorrentes da primeira execução.

6.4 Considerações Finais

Os resultados obtidos com a bateria de testes realizados através do *Phoronix Test Suite* revelam um cenário de equiparação entre o *VMware Workstation* e o *VirtualBox*. Tais resultados apontam que o nível de desempenho do *VirtualBox* pode ser considerado satisfatório, já que apresentou valores muito próximos ao do líder de mercado na grande maioria dos testes realizados. Outro aspecto importante revelado pelos experimentos aplicados sobre as ferramentas de virtualização foi o fraco desempenho apresentado pelo QEMU em relação aos seus concorrentes. Tal resultado foi surpreendente, já que a Tabela 1 do Anexo A apresentava um cenário de igualdade entre o *VirtualBox* e o QEMU (em conjunto com seu acelerador KQEMU), e definitivo para a desconsideração desta segunda solução pelo modelo aqui proposto.

A partir dos experimentos aplicados sobre a Camada de Gerenciamento WSE-OS, é possível avaliar o impacto da degradação gerada pela comunicação TCSC e sistema de virtualização no processo de inicialização de uma imagem de SO, já somado o tempo de *boot* do *middleware* e considerando otimização em memória *Flash Driver* USB. Tais resultados apresentam que o processo completo possui uma latência adicional de apenas 5,82 segundos comparado a uma inicialização nativa em *hardware* de mesmo cliente, representando uma degradação inferior a 13%.

Outro importante aspecto indicado pelos testes foi o ganho de desempenho do sistema através do uso de *cache*. O uso deste mecanismo em memória *Flash* garante que, após o primeiro *boot* do sistema WSE-OS, o processo de carregamento de um SO virtualizado no servidor tenha um rendimento 16,31% superior.

Os experimentos realizados nesta seção também demonstram que o modelo de virtualização centralizada proposto pela solução WSE-OS é válido para cenários onde o limite de memória RAM utilizada pelos sistemas operacionais convidados não ultrapassa 50% da quantidade disponível no servidor, sendo que nesses casos, a degradação do sistema operacional não inviabiliza a utilização do ambiente.

7. CONCLUSÕES

Este estudo apresentou que a combinação das técnicas virtualização, acesso remoto, ambiente de trabalho remoto e sistema de arquivos possibilita a criação de uma solução de gerenciamento de ambientes computacionais sem fio através de um conjunto de ISUs, sem com isso degradar o desempenho do ambiente em função do meio de comunicação (rede sem fio) ou da quantidade de sistemas operacionais virtualizados executando concorrentemente no servidor de forma a inviabilizar esta solução. Contudo, o desafio consiste em oferecer todos os benefícios da centralização do gerenciamento e da mobilidade das redes sem fio em um sistema que seja otimizado, escalável e transparente o suficiente a ponto de não comprometer a usabilidade do sistema.

Por ser um modelo baseado em infra-estrutura virtuais de *desktops* (VDI), esta construção concede os benefícios do gerenciamento centralizado sem que o enlace de comunicação sem fio seja degradado. Os testes realizados sobre o ambiente utilizado por este projeto mostram que a degradação de desempenho para a instanciação de um SO de referência é inferior a 13% do tempo de execução nativa em máquina cliente para uma conexão ativa. Além disso, os experimentos confirmam uma alta escalabilidade do ambiente, suportando, para a configuração de *hardware* utilizada, um número de até quarenta clientes sobre um único servidor e através de um único ponto de acesso à rede. Para ambientes com diferentes características, este número varia em função da quantidade de memória apresentada pelo servidor, da sua capacidade de processamento e da taxa de transmissão apresentada pela rede sem fio.

Mesmo introduzindo certa degradação no sistema em função do meio de comunicação, da virtualização e do compartilhamento dos recursos disponíveis no servidor, o WSE-OS mostra-se uma alternativa viável de execução centralizada, pois é capaz de oferecer os benefícios do gerenciamento centralizado mesmo em ambientes onde os clientes apresentam configuração heterogênea de *hardware*. Além disso, com a evolução do poder computacional dos novos processadores e das taxas de transmissão das redes sem fio, esta degradação de desempenho tende a ser cada vez menor, permitindo que o modelo de gerenciamento através da virtualização centralizada do WSE-OS possa oferecer todas as

vantagens apresentadas neste estudo, porém com eficiência mais próxima do desempenho nativo de um computador.

Tecnologias de virtualização atuais, como os modelos de virtualização com auxílio por *hardware*, permitem que uma solução como a WSE-OS, baseada em virtualização centralizada, apresente uma menor degradação de desempenho, contribuindo também para que o rendimento dos sistemas operacionais fique compatível com a execução nativa. Isto significa oferecer aos computadores clientes a possibilidade de execução de um sistema operacional virtualizado com desempenho próximo ao do mesmo executando diretamente sobre o processador.

Atendendo a outra característica desejável, o modelo WSE-OS utiliza apenas ferramentas gratuitas e uma infra-estrutura de comunicação de baixo custo (rede sem fio) e que permite mobilidade. Isso torna possível a adoção da solução por organizações dos mais variados portes, partindo de médias empresas, escolas, laboratórios de informática, onde as configurações de servidores são modestas, até grandes corporações, onde há servidores avançados e que permitem um melhor desempenho e um número maior de usuários. Junto com a redução de custos de infra-estrutura obtém-se também a redução de custos com manutenção do ambiente por parte dos gerenciadores de TI.

Este modelo ainda pode, com poucas modificações, ser adaptado para o contexto da internet, pois como visto, a transferência de dados se dá de maneira otimizada, reduzindo a taxa de transmissão. Isto permite que ambiente alcance tempos de resposta muito baixos, evitando que a banda da internet venha a ser motivo de gargalo e demora no sistema.

O WSE-OS é uma solução gratuita de gerenciamento centralizado que agrega os benefícios descritos, surgindo como alternativa inédita aos modelos existentes no mercado, pois, ao contrário destes, é especificamente estruturado sobre meio de comunicação sem fio.

7.1 Contribuições deste Trabalho

A principal contribuição deste trabalho volta-se para a proposta e validação de um modelo de gerenciamento centralizado de computadores em ambiente sem fio. Os estudos

realizados para o desenvolvimento da Camada de Gerenciamento WSE-OS possibilitaram a criação de tal solução utilizando a abordagem de *Virtual Desktop Infrastructure* (VDI) por meio de redes sem fio, contribuindo para pesquisas acadêmicas futuras. Além do modelo de gerenciamento, os seguintes pontos são contribuições deste trabalho:

- Um estudo bibliográfico sobre as principais tecnologias de ambientes de trabalho remoto, virtualização e gerenciamento centralizado de computadores;
- Proposta de uma arquitetura modular baseada em ambiente de trabalho remoto aliado a um mecanismo de virtualização, viabilizando a utilização de ISUs no modelo de VDI em ambientes dotados de redes com taxas de transmissão limitadas;
- Uma análise de desempenho de três das mais conhecidas soluções de virtualização: *VirtualBox*, *VMware Workstation* e QEMU com módulo de aceleração, para a aplicação desenvolvida.

7.2 Trabalhos Futuros

Como indicativo para continuidade deste trabalho são citados:

- Adequação e validação da solução WSE-OS para funcionamento através da Internet, permitindo assim que seus usuários possam utilizar qualquer computador como cliente ao redor do mundo;
- Aprimoramento no mecanismo de acesso a dispositivos removíveis de forma a permitir que os usuários interajam com aqueles como se estivessem sido conectados a um computador local dotado de um sistema operacional nativo, evitando o processo de navegação pelas pastas compartilhadas;
- Criar um mecanismo que permita a transmissão de áudio, de forma eficiente, para que os clientes possam se utilizar de aplicativos multimídias;
- Criar uma ferramenta de análise do ambiente, para que administradores de rede consigam visualizar resultados, em *real-time*, a respeito do funcionamento do ambiente, como consumo de memória, utilização de CPU, I/O de disco, I/O de rede, entre outras, para assim poderem tirar conclusões e considerarem alterações específicas para o caso em análise;

- Estender o funcionamento da solução WSE-OS para dispositivos móveis, permitindo que usuários tenham acesso a um sistema operacional remoto através de um *Smartphone*, PDA, Celular, etc;
- Estender o funcionamento da solução WSE-OS para oferecer aos clientes acesso a apenas aplicações virtualizadas no servidor WSE-OS, deixando o modelo WSE-OS mais completo de forma a abranger todas as necessidades impostas pela diversidade de ambientes existentes em corporações e instituições de ensino, por exemplo;
- Desenvolvimento de um sistema de equilíbrio de carga automatizado que seja capaz de monitorar os recursos dos servidores reais, melhorando o uso destes recursos a fim de evitar que algum servidor fique sobrecarregado enquanto outros estão subutilizados. Desta forma seria possível o uso de um conjunto de servidores replicados possibilitando melhores níveis de desempenho e escalabilidade do sistema WSE-OS.

REFERÊNCIAS BIBLIOGRÁFICAS

ADAMS, K., AGESEN, O. **A Comparison of Software and Hardware Techniques for x86 Virtualization**. Proceedings of the 12th international conference on Architectural support for programming languages and operating systems (ASPLOS). 2006. San Jose, Califórnia, 2006, p. 2-13.

AGUIAR, C. S. et al. **A Tool to Simplify the Management of Homogeneous and Heterogeneous Grids**. Proceedings of the IEEE Symposium on Computers and Communications (ISCC). 2009. Sousse, Tunisia, 2009, p. 295-298.

ALKMIM, G. P. **Aplicações da Virtualização em Empresas**. 2009. II Workshop de Informática Aplicada e Desenvolvimento de Jogos para Computadores e Dispositivos Móveis. UNIFESO - Teresópolis, 2009.

AMD. **AMD Virtualization**. 2010. Disponível em: <http://www.amd.com/br/products/technologies/virtualization/Pages/virtualization.aspx>. Acesso em: 14 abr. 2010.

ANDRADE, M. T. **Um estudo comparativo sobre as principais ferramentas de virtualização**. 2006. Monografia (conclusão de curso) - Universidade Federal de Pernambuco, Centro de Informática, Recife.

ARAUJO, A. M. G. **XDMCP**. 2006. Disponível em: < http://www.slideshare.net/hudsonaugusto/xdmcp?src=embed>. Acesso em: 17 set. 2009.

BAPTISTA, V. A. **Geocolab**. 2008. Monografia (conclusão de curso) – Universidade Federal do Espírito Santo, Vitória.

BARATTO, A. R.; KIM, N. L.; NIEH, J. **THINC: A Virtual Display Architecture for Thin-client Computing**. Proceedings of the twentieth ACM Symposium on Operating Systems Principles. 2005. Brighton, United Kingdom, 2005, p. 277 – 290.

BARHAM, P. et al. **Xen and de Art of Virtualization.** Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003. Bolton Landing, New York, USA, 2003.

BARUCHI, J. H. **Comparativo entre Ferramentas de Virtualização**. 2008. Monografia (conclusão de curso) - Faculdade de Jaguariúna, Jaguariúna.

BELLARD, F. **QEMU, a fast and portable Dynamic Translator**. Proceedings of the annual conference on USENIX Annual Technical Conference. 2005. Anaheim, Califórnia, 2005, p. 41.

BJERKE, H. K. F. **HPC Virtualization with Xen on Itanium.** 2005. Tese (Mestrado) - Norwegian University of Science and Technology, Trondheim.

CARISSIMI, A. **Virtualização: da teoria a soluções**. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2008, Rio de Janeiro. Livro texto dos minicursos. Rio de Janeiro: SBC, 2008. p. 174-199.

CHAO, T. **GNU/Linux XDMCP HowTo**. 2003. Disponível em: <http://tldp.org/HOWTO/XDMCP-HOWTO/>. Acesso em: 27 jul. 2009.

CHERVENAK, A. et al. **The data Grid: Towards an architecture for the distributed management and analisys of large scientific datasets**. Journal of Network and Computer Applications. Academic Press, Vol. 23, Issue 3, p. 187-200, jan. 2000.

COOPERSMITH, A. **X.Org HOME**. 2010. Disponível em: <http://www.x.org/wiki/Home>. Acesso em: 05 abr. 2010.

CREPALDI, L. G. **Middleware de Comunicação entre Objetos Distribuídos para Gerenciamento de Computadores baseado em Redes Sem Fio (WSE-OS)**. 2011 (a ser publicado). Tese (Mestrado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru.

CRUZ, D. I. **FLEXLAB: Middleware de virtualização de hardware para gerenciamento centralizado de computadores em rede**. 2008. Tese (Mestrado) - Universidade Estadual Paulista "Júlio de Mesquisa Filho", Bauru.

DARTWARE. **InterMapper Flows**. 2011. Disponível em: <http://www.intermapper.com/products/intermapper-flows/features> Acesso em: 05 dez. 2010.

FARHAT, C.; ELIAN, S. **Estatística Básica**. São Paulo: LCTE Editora, 2008. 240p.

GOLDBERG, R. P. **Architectural Principles for Virtual Computer Systems**. 1973. Tese (Doutorado) - Harvard University, HQ Electronics Systems Division, Bedford, Massachusetts.

GOOGLE. **NeatX**. 2010. Disponível em: <http://code.google.com/p/neatx>. Acesso em: 19 out. 2010.

GUEDES, R. M.; SILVA, E. M. **Introdução ao Uso do Linux v. 4**. 2006. Disponível em: <http://www.lee.eng.uerj.br/~elaine/introducao-ao-uso-do-linux.pdf>. Acesso em: 07 out. 2009.

HABIB I. **Virtualization with KVM**. Linux Journal, Seattle, v. 2008, n. 166, p. 8, fev. 2008.

HENRY, M. **PXE Manageability Technology for EFI**. Intel Developer Update Magazine, Intel., p. 3, oct. 2000. Disponível em: <http://www.intel.org/technology/magazine/systems/it10004.pdf>. Acesso em: 11 mar. 2010.

HOSKINS, M. E. **SSHFS: super easy file access over SSH**. Linux Journal, Seattle, v. 2006, n. 146, p. 4, jun. 2006.

INTEL. **Preboot Execution Environmet (PXE) Specification Version 2.1**. 1999. Disponível em <http://www.pix.net/software/pxeboot/archive/pxespec.pdf>. Acesso em: 01 abril 2010.

INTEL. **Intel Virtualization Technology**. 2010. Disponível em: <http://www.intel.com/technology/virtualization/technology.htm?iid=tech_vt+tech>. Acesso em: 29 abril 2010.

JONES, M. T. **Virtual Linux - IBM Developer's Works**. 2006. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-linuxvirt/?ca=dgr-lnxw01>. Acesso em: 15 mar. 2010.

JONES, M. T. **Emulação do Sistema com o QEMU**. 2006. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-linuxvirt/?ca=dgr-lnxw01>. Acesso em: 15 mar. 2010.

KDE. **Welcome to KDE UserBase**. 2010. Disponível em: <http://userbase.kde.org/>. Acesso em: 16 nov. 2010.

KING, S. T. et al. **Operating System Support for Virtual Machines**. Proceedings of the USENIX Annual Technical Conference. 2003. Berkeley, Califórnia, 2003, p. 71-84.

KIVITY, A. et al. **kvm: the Linux virtual machine monitor**. 2007. Proceedings of the Linux Symposium. Ottawa: Linux Symposium Inc., 2007. v. 1, p. 225–230.

KOJIMA, A. K. **Window Maker - User's Guide**. 2011. Disponívem em: <http://windowmaker.org/documentation.php?show=userguide>. Acesso em: 02 fev. 2011.

LAUREANO, M. **Máquinas Virtuais e Emuladores: Conceitos, Técnicas e Aplicações**. São Paulo: Novatec, 2006. 184 p.

LUTZ, M. **Programming Python**. 2nd ed. Sebastopol: O'Reilly Media, 2001. 1296 p.

MATTOS, D. M. F. **Virtualização: VMWare e Xen**. GTA/POLI/UFRJ. 2008. Disponível em: <http://www.gta.ufrj.br/grad/08_1/virtual/artigo.pdf>. Acesso em: 02 fev. 2010.

MEYER, U.; WETZEL S. **A man-in-the middle attack**. Proceedings of the ACM Workshop on Wireless Security. 2009. Philadelphia, USA, 2009, p. 90 – 97.

MICROSOFT. **Windows Virtual PC**. 2011. Disponível em: <http://www.microsoft.com/windows/virtual-pc/default.aspx>. Acesso em: 18 jan. 2011.

MORGADO, E.; CRUZ, D. I.; TWANI, E.; **Paving the Way for a Dynamic and Mature ICT Infrastructure in Education: A Case for Schools in Emerging Markets**. 2008. Proceedings of

the International Conference on Engineering and Technology Education - INTERTECH 2008. Santos, São Paulo. 2008, p. 79.

MORIMOTO, C. E. **Redes e Servidores:** Guia Prático. 2. ed. Porto Alegre: Sul Editores, 2006. 448 p.

MORIMOTO, C. E. **Servidores Linux:** Guia Prático. Porto Alegre: Sul Editores, 2008. 735 p.

MORIMOTO, C. E. **XDMCP**. 2009a. Disponível em: <http://www.hardware.com.br/termos/xdmcp>. Acesso em: 25 nov. 2009.

MORIMOTO, C. E. **KQEMU: QEMU 5X mais rápido. Um concorrente à altura do VMware**. 2009b. Disponível em: <http://www.guiadohardware.net/artigos/kqemu/>. Acesso em: 06 mar. 2010.

OLIVEIRA, A. B. et al. **Acesso Remoto à Computadores**. 2006. Disponível em: <http://www.cesarkallas.net/arquivos/faculdade/topicos2/acesso-remoto.pdf>. Acesso em: 21 ago. 2009.

ORACLE CORPORATION. **Oracle VM Virtualbox, User Manual**. 2011. Disponível em: <http://dlc.sun.com.edgesuite.net/virtualbox/4.0.2/UserManual.pdf>. Acesso em: 05 fev. 2011.

PHORONIX MEDIA. **Phoronix Test Suite**. 2010. Disponível em: <http://phoronix-test-suite.com/>. Acesso em: 02 maio 2010.

PHORONIX MEDIA. **BYTE Unix Benchmark 3.6**. 2009. Disponível em: <http://tests.phoronix-test-suite.com/index.php?q=byte>. Acesso em: 05 nov. 2010.

PHORONIX MEDIA. **Getting Started**. 2011. Disponível em: <http://www.phoronix-test-suite.com/?k=documentation>. Acesso em: 03 nov. 2010.

POPEK, G.; GOLDBERG, R. **Formal Requirements for Virtualizable Third Generation Architectures**. Comunications of the ACM, New York, v. 17, n. 7, July 1974.

PUEL. **VirtualBox Personal Use and Evaluation License (PUEL)**. [S.l.]. 2008. Disponível em: <http://www.virtualbox.org/wiki/VirtualBox_PUEL>. Acesso em: 19 out. 2010.

QEMU. **About QEMU**. 2010. Disponível em: < http://wiki.qemu.org/Index.html>. Acesso em: 30 mar. 2010.

REAL, L. C. V. **Uma Arquitetura para Análise de Fluxo de Dados Estruturados Aplicada ao Sistema Brasileiro de TV Digital**. 2009. Tese (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo.

REALVNC. **Frequently Asqued Questions**. 2009. Disponível em: <http://www.realvnc.com/support/faq.html>. Acesso em: 25 out. 2009.

REGIS, S. **Getting Started with NX**. 2007. Disponível em: <http://www.nomachine.com/documents/getting-started.php>. Acesso em: 26 out. 2009.

RICHARDSON, T. **The RFB Protocol**. 2000. Disponível em: <http://www.realvnc.com/docs/rfbproto.pdf>. Acesso em: 28 out. 2009.

ROSENBLUM, M. **The Reincarnation of Virtual Machines**. ACM Queue, New York, v. 2, n. 5, july/aug. 2004.

SILVA, I. C. S. **Sistema de Visualização de Dados para o Centro de Pesquisas Metereológicas da UFPel**. 2000. Monografia (conclusão de curso) – Universidade Federal de Pelotas, Pelotas.

SIRER, E. G. et al. **Distributed Virtual Machines: A System Architecture for Network Computing**. Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications. 1998. Sintra, Portugal, 1998, p. 13-16.

SISOFTWARE. **SANDRA Software Benchmark**. 2010. Disponível em: < http://www.sisoftware.net > Acesso em: 16 out. 2010.

SMITH, J.; NAIR, J. **The architecture of Virtual Machines**. Journal Computer, Los Alamitos, CA, USA, v. 38, n. 5, p. 32-38, may 2005.

SOUSA, S. H. G. **Suporte Remoto: VNC**. Revista PC's, ed. 47, p. 62-63, fev. 2004.

SUN MICROSYSTEMS. **Sun VirtualBox User Manual**. [S.l.]. 2010. Disponível em: <http://download.virtualbox.org/virtualbox/3.1.6/UserManual.pdf>. Acesso em: 15 out. de 2010.

TEIXEIRA, R. A. **SW-V: Modelo de streaming de software baseado em técnicas de virtualização e transporte peer-to-peer**. 2010. Tese (Mestrado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru.

THE GNOME PROJECT. **GNOME: The Free Software Desktop Project**. 2010. Disponível em: < http://www.gnome.org/>. Acesso em: 20 nov. 2010.

THE GTK+ TEAM. **The GTK+ Project**. 2010. Disponível em: http://www.gtk.org/. Acesso em: 20 nov. 2010.

VIRTUALBOX. **VirtualBox**. 2010. Disponível em: <http://www.virtualbox.org/wiki/VirtualBox>. Acesso em: 17 out. 2010.

VMWARE. **VMware ESX Server – User's Manual**. 2002. Disponível em: <http://www.vmware.com/pdf/esx_15_manual.pdf>. Acesso em: 14 fev. de 2010.

VMWARE. **Virtual Desktop Infra-structure**. 2007. Disponível em: <http://www.vmware.com/pdf/virtual_desktop_infrastructure_wp.pdf>. Acesso em: 17 fev. de 2010.

VMWARE. **VMware Server**. 2011a. Disponível em: < http://www.vmware.com/products/server/>. Acesso em: 11 jan. de 2011.

VMWARE. **VMware Workstation**. 2011b. Disponível em: <http://www.vmware.com/pdf/virtual_desktop_infrastructure_wp.pdf>. Acesso em: 11 jan. de 2011.

VMWARE. **VMware View 4.5**. 2011c. Disponível em: <http://www.vmware.com/br/products/view/ >. Acesso em: 03 jan. de 2011.

WALDSPURGER, C. A. **Memory Resource Management in VMWare ESX Server**. 2002. Proceedings of the 5th symposium on Operating systems design and implementation. New York, New York, 2002, p. 181-194.

XEN SOURCE. **Xen Source**. 2010. Disponívem em: <http://www.xensource.com>. Acesso em: 18 fev. 2010.

YLONEN, T. **The Secure Shell (SSH) Protocol Architecture**, RFC 4251, 2006a. Disponível em: <http://tools.ietf.org/html/rfc4251>. Acesso em: 12 set. 2010.

YLONEN, T. **The Secure Shell (SSH) Connection Protocol**, RFC 4254, 2006b. Disponível em: <http://tools.ietf.org/html/rfc4254>. Acesso em 19 set. 2010.

ZHOU, Y.; ZHANG, Y.; XIE, Y. **Virtual Disk based Centralized Management for Enterprise Networks**. Proceedings of the 2006 SIGCOMM workshop on Internet network management. 2006. Pisa, Italia, 2006, p. 23.

ANEXO A

Análise sobre as ferramentas de virtualização.

As soluções para virtualização consideradas na implementação da Camada de Gerenciamento WSE-OS foram o QEMU com o acelerador KQEMU, o *Xen*, o KVM e o *VirtualBox*, todas elas possuindo licenciamento aberto ou para fins de pesquisa. Embora todas estas ferramentas tenham sido analisadas dentro do contexto da solução WSE-OS, determinadas características técnicas de implementação dos *software* inviabilizam o uso de algumas soluções de acordo com as exigências estabelecidas na seção 5.4.

De todas as ferramentas consideradas, as únicas que se enquadram perfeitamente em todos os requisitos estabelecidos pela Camada de Gerenciamento WSE-OS são as soluções *VirtualBox* e QEMU com o módulo de aceleração KQEMU, sendo a primeira uma alternativa mais moderna e completa, como pode ser visto na seção 3.7.6, atendendo também a todas as características desejáveis, conforme apresentado na seção 5.4, para a ferramenta de virtualização. A Tabela 1 faz um comparativo entre os *software* de virtualização considerados neste estudo.

Tabela 1 - Comparativo entre as ferramentas de virtualização. Fonte: (CRUZ, 2008).

	Emulação	Tradução Binária	Assistência por *Hardware*	Paravirtualização
Compatibilidade com processadores (Exige extensões de virtualização?)	Excelente	Excelente	Fraca	Excelente
Compatibilidade de SO (Suporte a SO não modificado)	Excelente	Excelente	Excelente	Fraca
Desempenho	Fraco	Muito Bom	Excelente	Excelente
Sofisticação da VMM	Alta	Alta	Mediana	Mediana
Exemplos (Mvs com licença LGPL e/ou GPL)	QEMU	QEMU + KQEMU e VirtualBox (PUEL)	XEN e KVM	XEN
Exemplos (MVs com licença proprietária)		VMware, VirtualBox e Virtual PC	VMWare, XenApp e Virtual PC	XenApp

Das ferramentas de virtualização inicialmente consideradas para uso na Camada de Gerenciamento WSE-OS, foram descartadas todas as soluções que possuam licenciamento

proprietário, por não ser possível inseri-las no módulo de Mecanismo de Virtualização sem infringir os direitos de propriedade intelectual e industrial.

Analisando as soluções não proprietárias que possuem melhor desempenho, tem-se o *Xen*, que permite dois tipos de virtualização: a Paravirtualização e a Virtualização Completa. No primeiro caso, a inviabilidade está no fato de o *Xen* exigir um SO convidado modificado, e no segundo caso, o problema está no fato de o *Xen* exigir processadores que tenha extensões de virtualização em sua arquitetura (VT-x e AMD-V) e sua escolha limitaria a solução WSE-OS somente a servidores que possuam tais extensões.

Similarmente ao Xen, o sistema de virtualização adotado pela KVM também realiza virtualização completa se utilizando de extensões de virtualização, não atendendo aos requisitos estabelecidos anteriormente.

O aplicativo de virtualização QEMU com o acelerador KQEMU é uma solução gratuita e que realiza a virtualização completa (sem a necessidade de alterações nos SOs a serem virtualizados) e não exige que o processador possua extensões de virtualização (como as tecnologias VT-x e AMD-V). Além disso, tal solução também apresenta versão para execução sobre SO *Linux* e permite a execução de uma grande gama de sistemas operacionais convidados, como visto na seção 3.7.4.

A ferramenta *VirtualBox*, como já citado na seção 3.7.6, apresenta muitas semelhanças com a solução QEMU em conjunto com o KQEMU, pelo fato de o desenvolvimento da primeira ter sido baseada na segunda. Consequentemente, o aplicativo *VirtualBox* também realiza a virtualização completa não exigindo extensões de virtualização no processador. Como diferenciais do QEMU, o *VirtualBox*, apesar de não ter como obrigatório, também possibilita o uso das extensões de virtualização e oferece suporte a sistemas convidados 64-bit, sendo que no primeiro tal característica ainda é experimental. Outra funcionalidade encontrada no *VirtualBox* são os Adicionais para Convidado que, como explicado na seção 3.7.6, proporciona melhor integração entre sistema hospedeiro e sistema convidado.

Além de o *VirtualBox* ser uma solução mais completa do que a QEMU em conjunto com seu acelerador (KQEMU), existem ainda dois aspectos que tornam a segunda solução menos apropriada para o WSE-OS:

- O módulo KQEMU não está mais recebendo o foco no desenvolvimento que por sua vez foi transferido para o KVM;
- O QEMU apresentou um desempenho inferior ao *VirtualBox* em todos os testes apresentados na seção 6.2.

A partir desta análise foi definido a ferramenta *VirtualBox* para ser integrante do módulo de Mecanismo de Virtualização, pois além de atender a todos os requisitos e características desejáveis listados na seção 5.4, apresentou um desempenho melhor do que o QEMU (associado ao KQEMU) e muito próximo ao da solução desenvolvida pela empresa líder de mercado, VMware, podendo assim ser considerado como um nível satisfatório.