Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Um Arquitetura para Gerência de Rede de Máquinas Virtuais com ênfase na Emulação de Sistemas Distribuídos

Mauro Strelow Storch

Porto Alegre, 2008

Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Uma Arquitetura para Gerência de Rede de Máquinas Virtuais com ênfase na Emulação de Sistemas Distribuídos

Mauro Strelow Storch

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Prof. Dr. César A. F. De Rose

Porto Alegre
2008

Dados Internacionais de Catalogação na Publicação (CIP)

S884a Storch, Mauro Strelow.
Uma arquitetura para gerência de rede de máquinas virtuais com ênfase na emulação de sistemas distribuídos / Mauro Strelow Storch. – Porto Alegre, 2008.
83 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. César A. F. De Rose

1. Informática. 2. Arquitetura de Redes. 3. Gerência de Redes. 4. Máquinas Virtuais. 5. Sistemas Distribuídos. I. Título.

CDD 004.65

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "***Uma Arquitetura para Gerência de Rede de Máquinas Virtuais com Ênfase na Emulação de Sistemas Distribuídos***", apresentada por Mauro Strelow Storch, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 26/03/08 pela Comissão Examinadora:

Prof. Dr. César Augusto Fonticielha De Rose – Orientador — PPGCC/PUCRS

Prof. Dr. Luiz Gustavo Leão Fernandes – — PPGCC/PUCRS

Prof. Dr. Nicolas Maillard – — UFRGS

Homologada em 15../.10.../.2008., conforme Ata No. 021/08. pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@inf.pucrs.br
www.pucrs.br/facin/pos

Aos meus pais e minha irmã.

Agradecimentos

- Ao Professor César De Rose pelas orientações e a oportunidade de crescimento acadêmico;
- Ao Professor Adenauer Yamin por sua sempre presente dedicação e motivação;
- Aos colegas Rodrigo e Guilherme pelo companheirismo e pela dedicação no trabalho que juntos realizamos nesses quase dois anos de mestrado;
- Aos colegas do CPAD com os quais tive oportunidade de aprender muito;
- Aos meus pais e minha irmã que também sempre deram todo o apoio para realização desta etapa;
- À Hewlett-Packard pelo financiamento de meus estudos e ao incentivo a pesquisa.

Resumo

Pesquisas utilizando virtualização de hardware vêm sendo feitas em diversas áreas dentro da Ciência da Computação. A criação de ambientes virtuais com essa tecnologia pode ser implementada utilizando-se poucos recursos computacionais. Porém, a falta de ferramentas de configuração dificulta a criação de ambientes de larga escala. Recursos de rede, por exemplo, precisam ser gerenciados e configurados para atender às especificações do ambiente virtual que se deseja criar. Na intenção de facilitar esta gerência e reduzir os erros na configuração do ambiente virtual, apresentamos neste trabalho uma arquitetura de gerência de rede de máquinas virtuais. Dentre suas funções encontram-se questões como configuração de isolamento de rede e controle de vazão e latência. Na validação, aspectos de rede de um ambiente de emulação de sistemas distribuídos são configurados e uma aplicação é executada nesse ambiente na intenção de avaliar esses aspectos de comunicação.

Palavras-chave: Virtualização. Gerência de rede. Ambientes virtuais. Emulação de sistemas.

Abstract

Hardware virtualization researches are developed in different computer science areas. Virtual environments using virtualization technology can be created on few phisycal resources. However, the lack of configuration tools makes hard creation of large scale environments. Network resources, for instances, have to be managed and configured in order to follow the desired virtual environment especification. Aiming at making management easy as well as reducing configurations faults, we present in this work an architecture for virtual machine network management. Tasks of the architecture include issues such as configuration of network isolation and control of bandwidth and latency. In order to evaluate the architecture, a distributed system emulator was created over a set of virtual machines and an application was ran to analise the network communication.

Keywords: Virtualization. Network management. Virtual environments. System emulation.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CIM	*Common Information Model*	35
COW	*Cluster of Workstations*	57
DTD	*Document Type Definitions*	40
E/S	Entrada e Saída	28
HTTP	*HyperText Transfer Protocol*	35
JVM	*Java Virtual Machine*	30
MIB	*Management Information Base*	36
NetEm	*Network Emulator*	53
OID	*Object IDentifier*	36
SCP	*Secure Copy Protocol*	51
SNMP	*Simple Network Management Protocol*	34
TBF	*Token Bucket Filter*	51
XML	*eXtensible Markup Language*	55
VMM	*Virtual Machine Monitor*	26
WBEM	*Web-Based Enterprise Management*	34

Sumário

1 Introdução

Com o aumento do poder computacional das máquinas nos últimos anos, pesquisas utilizando tecnologias de virtualização de hardware surgem para explorar esse recurso através da execução de múltiplas máquinas virtuais em um mesmo computador. Dentro desse contexto, a possibilidade de criar ambientes virtuais de larga escala é facilitada. Porém, a configuração desse tipo de ambiente é difícil pela falta de ferramentas de suporte para atividade como a configuração e gerência de recursos como os de rede.

Questões como isolamento e controle de vazão e latência de rede entre máquinas virtuais são parâmetros que podem ser controlados por um administrador de um ambiente virtual. Esses aspectos precisam ser configurados em todas as máquinas do ambiente, porém, em um ambiente de larga escala, a configuração manual desses parâmetros demandaria muito tempo e também estaria sujeita a erros. Neste caso, uma ferramenta de configuração dos recursos de rede serviria como auxílio na criação de ambientes baseados em máquinas virtuais, diminuindo o trabalho dos administradores e reduzindo erros de configuração.

A observação de tal necessidade motiva o desenvolvimento deste trabalho que consiste em apresentar uma arquitetura para gerência e configuração de recursos de rede de máquinas virtuais. Essa arquitetura tem como objetivo agilizar o processo de configuração dos aspectos de rede de ambientes virtualizados. Ela consiste em elementos de descrição dos ambientes reais e virtuais, módulo de mapeamento de conexões de rede e estrutura de configuração distribuída.

Dentre os cenários que podem ser explorados utilizando este contexto, emulação de sistemas distribuídos foi o caso escolhido para validação da arquitetura. Esse tipo de sistema exige que as conexões entre as máquinas virtuais sejam configuradas de forma a se comportarem de diferentes modos, desde conexões locais de alta velocidade, até conexões de longa distância, com variação das taxas de vazão e latência.

Neste sentido, a arquitetura de gerência de rede de máquinas virtuais auxilia nas

configurações também nesse tipo de ambiente, atendendo aos requisitos recebidos do administrador do ambiente virtual.

1.1 Motivação e Objetivos

A criação de um ambiente distribuído com máquinas virtuais é uma alternativa onde com poucos recursos computacionais se pode criar um ambiente de larga escala. Além disso, é possível explorar questões como o controle de execução de aplicações, já que sistemas de virtualização oferecem nativamente recursos como monitoração e controle de máquinas virtuais.

A gerência dos recursos de rede deste ambiente precisa ser feita com o objetivo de aproximar o seu comportamento ao de ambientes reais. Questões como disponibilidade de nós e variação de vazão e latência de rede podem alterar a execução de uma aplicação, por isso é importante que estas e outras questões sejam avaliadas na intenção de minimizar seus problemas quando da execução em um ambiente real.

Dentro desse contexto será desenvolvido o principal foco deste trabalho, que apresenta uma arquitetura para configuração e gerência de redes de um ambiente baseado em virtualização. O principal objetivo dessa arquitetura é configurar as conexões de rede entre máquinas virtuais, previamente instaladas e inicializadas em um sistema, de forma a aproximar seu comportamento ao de um ambiente real.

No final deste trabalho, componentes de um ambiente distribuído real são instalados e configurados em um ambiente de máquinas virtuais na intenção de demonstrar e validar os aspectos de rede gerenciados pela arquitetura proposta.

1.2 Organização do Trabalho

Para melhor organização deste trabalho apresentam-se, nos dois próximos capítulos, os principais aspectos conceituais sobre virtualização e gerência de rede. O Capítulo 4 apresenta a descrição e implementação da arquitetura de gerência de rede para a configuração de sistemas distribuídos baseado em máquinas virtuais. Logo após, no Capítulo

5, é feita a avaliação da arquitetura com a implementação de um ambiente de *Grid*[1] formado por máquinas virtuais. Questões como isolamento, vazão e latência de rede foram medidos no intuito de demonstrar a efetividade da configuração feita pela arquitetura proposta. Para finalizar, o Capítulo 6 apresenta as conclusões e as considerações finais deste trabalho.

[1]Grid: o termo refere-se ao conceito de Grade Computacional

2 Virtualização

A tecnologia de virtualização é um conceito importante no desenvolvimento deste trabalho. Neste capítulo será feita a apresentação dos tipos de virtualização e do modelo que será utilizado neste trabalho.

Historicamente a virtualização nasceu nos anos 1960 quando a IBM disponibilizou esse recursos em seus *mainframes*, de forma a prover o acesso de clientes através de terminais. No *mainframe* cada terminal tinha a sua máquina virtual de forma independente. Assim, um único computador servia processamento para múltiplos clientes, dando a impressão para o usuário de estar utilizando uma máquina local real [1].

Durante os anos 1980 e 1990 a virtualização de hardware praticamente desapareceu, em função do aumento do poder computacional das estações e a redução de seu custo, ao ponto de não ser mais vantagem a aquisição de um supercomputador para atender estações de trabalho. Porém, surgiram outras tecnologias de virtualização, de mais alto nível, como as máquinas virtuais *Java* [2].

Com o aumento considerável da capacidade de processamento dos computadores nos últimos anos, novas pesquisas surgem para explorar este poder computacional utilizando a virtualização de hardware [3].

2.1 Modelos de virtualização

Um computador pode, basicamente, ser ilustrado como camadas dividas em quatro níveis sobrepostos horizontalmente : hardware, sistema operacional, bibliotecas e aplicações. A virtualização é uma camada intermediária que pode surgir em qualquer ponto dessa arquitetura, seja ela de mais alto nível, como as máquinas virtuais *Java*, ou de um nível mais baixo, como os virtualizadores de hardware. Em ambos os modelos a função principal é a abstração da camada inferior [4, 5]. Por exemplo, um programa

Java executa sobre a máquina virtual independentemente do Sistema Operacional que está executando sobre o hardware.

Para que se possam demonstrar algumas diferenças entre os níveis de aplicação da virtualização, nas próximas seções serão apresentadas características específicas de funcionamento de diferentes modelos de virtualização.

2.2 Nível de Hardware

O conceito de virtualização no nível de hardware vem sendo utilizado nos últimos anos na intenção de melhor aproveitar os recursos computacionais oferecidos pelos computadores. A virtualização do nível de hardware atua, basicamente, oferecendo suporte a execução de mais de um sistema operacional em um mesmo computador. Ela é responsável por gerenciar os recursos físicos e os compartilhar com as máquinas virtuais. A seguir é feita a apresentação do modelo de virtualização VMM.

2.2.1 VMM: Monitores de Máquinas Virtuais

Consideremos um ambiente de produção onde desenvolvedores precisam testar novas funcionalidades de um sistema. Em um ambiente convencional usa-se um computador para o desenvolvimento e outro, normalmente de menor capacidade, para os testes. Uma alternativa para este caso seria a utilização dos recursos de um Monitor de Máquinas Virtuais (VMM) , que nos possibilita a execução de duas máquinas virtuais em um mesmo computador de forma completamente independente. Assim, pode-se ter uma máquina virtual como ambiente de desenvolvimento e outra como ambiente de testes.

Basicamente um computador opera em modo *user* (com restritos privilégios para execução de instruções) e em modo *supervisor* (com total privilégio de execução de instruções). Em um ambiente de Máquinas Virtuais, o VMM opera em modo *supervisor* e controla todos os recursos físicos do hardware, compartilhando-os com as VMs que executam em modo *user*. O VMM escalona as Máquinas Virtuais de forma semelhante ao escalonamento de processos em Sistemas Operacionais e aloca ciclos de processa-

mento para cada uma delas [6]. Como as Máquinas Virtuais executam em modo *user*, nenhuma instrução executada por um Sistema Operacional convidado (executado em uma máquina virtual) acessa diretamente o hardware. O VMM é o responsável pela execução de instruções de hardware: ele intercepta e trata essas instruções para garantir a segurança e a integridade do ambiente. Assim, por exemplo, se uma Máquina Virtual recebe do seu Sistema Operacional a instrução de *halt*, o VMM irá simplesmente desativar a Máquina Virtual que originou a instrução e continuará a execução das demais VMs.

Essa intervenção do VMM é a causa do *overhead* desse tipo de sistema. Mas podemos realçar algumas vantagens da virtualizaçao como:

- *Segurança:* execução de ambientes com níveis de segurança diferentes em Máquinas Virtuais diferentes;
- *Confiabilidade:* a falha de um software em uma Máquina Virtual não altera o funcionamento das demais;
- *Custo:* é possível a redução de custos implementando pequenos servidores em um mesmo computador;
- *Balanceamento de carga:* pode-se migrar Máquinas Virtuais para computadores de poder computacional mais elevado.

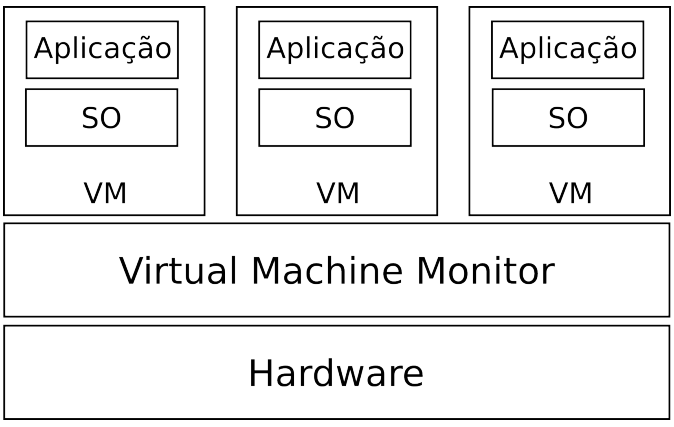

Figura 1 – Arquitetura de Monitores de Máquinas Virtuais

Pode-se visualizar, na Figura 1, um exemplo de como a virtualização atua em um ambiente com várias Máquinas Virtuais. Sobre a camada mais inferior, que se refere ao hardware físico, encontra-se o Monitor de Máquinas Virtuais que nos oferece suporte à execução de múltiplas Máquinas Virtuais, cada uma com seu Sistema Operacional, compartilhando os recursos físicos de um mesmo computador.

Técnicas de escalonamento, como *time-sharing*, aplicadas em Sistemas operacionais são também utilizadas pelos VMMs para o compartilhamento de recursos hardware entre as Máquinas Virtuais. Desta forma cabe uma revisão da virtualização de recursos de hardware por Monitores de Máquinas Virtuais, que é feita a seguir.

2.2.2 Virtualização de Recursos

Nos *mainframes* da década de 1960 o sistema de E/S (Entrada e Saída) era baseado em uma arquitetura onde os dispositivos eram acessados por diferentes canais pelo processador. Assim, as Máquinas Virtuais podiam fazer E/S diretamente no dispositivo sem causar danos ao ambiente e, por isso, o *overhead* desse tipo de operação era baixo [7].

Atualmente, a arquitetura dos computadores evoluiu e um grande conjunto de dispositivos pode ser associado a eles. Por isso se tornou mais complicado manter a segurança na virtualização, sendo necessário que o VMM gerencie esses diferentes dispositivos causando, assim, um *overhead* na performance. Além disso, alguns dispositivos como placas gráficas e interfaces de rede de alta velocidade necessitam de alto desempenho do ambiente, causando situações críticas na virtualização como a necessidade de utilização de novas técnicas de escalonamento do processador para suprir essa demanda.

Mesmo em ambiente convencionais a utilização de recursos físicos que dependem de E/S é sempre causa de *overhead* no desempenho do sistema. Em um ambiente virtualizado o problema relacionado ao desempenho se agrava porque a camada intermediária entre as instruções e os dispositivos trata questões de segurança e consistência para garantir, por exemplo, o isolamento das Máquinas Virtuais. Esta camada, chamada de VMM, é responsável por verificar a integridade das instruções de hardware, executá-las e devolver uma resposta para a Máquina Virtual. O seu funcionamento varia de acordo com o tipo de dispositivo podendo, em alguns casos, não afetar o desempenho significativamente.

Pode-se tomar como exemplo o acesso de Máquinas Virtuais à memória, onde cada máquina virtual recebe uma faixa de endereços e a permissão de leitura e escrita. Neste caso, o VMM precisa somente garantir que não haverá interrupções fora da faixa de endereços delegada a cada Máquina Virtual.

Em muitos modelos de dispositivos é preciso um gerenciamento bem restrito das requisições. O caso dos dispositivos de rede possui algumas características particulares de virtualização que serão abordadas no Capítulo 4.

2.3 Nível de Sistema Operacional

Conhecido da literatura como *Operating system-level virtualization*, este modelo de virtualização é utilizado em ambientes onde o *kernel* da máquina real e da máquina virtual é o mesmo. Um pouco diferente do modelo de virtualização anterior, onde era preciso ser feita a instalação de todo um sistema com *kernel* modificado na Máquina Virtual, neste modelo é feito apenas o isolamento dos processos, mantendo a instalação do sistema idêntico em todas as máquinas virtuais. Dependendo do virtualizador utilizado, ele pode oferecer diferentes recursos de isolamento. Na Tabela 1[1] pode-se perceber que sistemas como OpenVZ [8] e Virtuozzo [9] oferecem total isolamento para as instâncias de Máquinas Virtuais.

Tabela 1 – Funcionalidades de virtualizadores no nível de sistema operacional.

Sistema Virtualizador	Sistema Operacional	Recursos						
		1	2	3	4	5	6	7
chroot	UNIX	×						
FreeVPS	Linux	×	×		×	×	×	
Linux-VServer	Linux	×	×	×	×	×	×	
OpenVZ	Linux	×	×	×	×	×	×	×
SWsoft Virtuozzo	Linux, Windows	×	×	×	×	×	×	×
FreeBSD Jail	FreeBSD	×					×	

1-Isolamento do sistema de arquivos 2-Cota de Disco 3-Limitação de E/S 4-Limitação de memória 5-Cota de CPU 6-Isolamento de Rede 7-Migração e *checkpointing*

[1] Adaptado de `http://en.wikipedia.org/wiki/Jail_(computer_security)`

Neste conceito de virtualização o principal objetivo é oferecer recursos, como sistema operacional, sistema de arquivos, arquivos de configuração e bibliotecas, de forma isolada – se diferenciando do modelo anterior apenas quanto às Máquinas Virtuais (máquinas convidadas), que utilizam o mesmo *kernel* da máquina hospedeira.

2.4 Nível de Linguagem de Programação

O conceito de virtualização no nível de linguagem de programação vem sendo, nos últimos anos, fortemente introduzido no mercado com as Máquinas Virtuais Java [2] (JVM, do inglês *Java Virtual Machine*) .

O principal objetivo deste modelo de virtualização é oferecer uma plataforma de execução para aplicações, de forma que a arquitetura do hardware e o sistema operacional não tenham influência sobre estas, ou seja, um ambiente para aplicações multiplataforma.

Este modelo de virtualização introduz uma camada de abstração entre a aplicação e o sistema operacional. Essa camada é responsável por interpretar e invocar as chamadas de sistema feitas pela aplicação. Assim, independentemente do sistema operacional que esteja sob a esta camada de virtualização, a aplicação será sempre a mesma. A única mudança será da própria camada de abstração, que deverá se adaptar de acordo com o sistema operacional [10].

Assim como nos outros modelos de virtualização, a camada de abstração causa - *overhead* na execução da aplicação, no entanto, oferece o recurso de portabilidade da aplicação para diferentes sistemas operacionais e arquiteturas de hardware.

Além da máquina virtual Java existem no mercado outros virtualizadores de nível de linguagem de programação como o .NET CLI [11].

2.5 Nível de Biblioteca

Na maioria das aplicações, a utilização de bibliotecas de programação permite a adaptação de um programa a um determinado ambiente abstraindo alguns aspectos es-

pecíficos. Essa abstração é oferecida através de interfaces de programação, também conhecida na literatura como API (*Application Programming Interface*).

A virtualização, no caso de bibliotecas, não acontece como nos outros níveis onde há isolamento de processos ou compartilhamento de recursos. Neste caso a vantagem está na possibilidade de desenvolver uma aplicação que, ao ser portada para outros ambientes, precisar ser apenas recompilada com bibliotecas específicas, não sendo necessária a modificação do seu código fonte. Isso pode ser um fator relevante quando se trata de aplicações de grande porte que possuem códigos extensos.

2.6 Considerações deste Capítulo

A virtualização é uma tecnologia que pode ser aplicadas em diferentes níveis de sistemas de computação. Através de exemplos comuns no mercado como as Máquinas Virtuais Java, pôde-se apresentar como se aplicam os conceitos de virtualização. No entanto, a virtualização de hardware teve uma maior ênfase neste capítulo, pois, seus conceitos e funcionalidades serão discutidos e utilizados no desenvolvimento deste trabalho.

No próximo capítulo, ainda sobre os conceitos tecnológicos aplicados a este trabalho, serão discutidos e apresentados conceitos e características de gerência de rede. Aspectos como funcionalidades e ambientes de aplicação concluem o embasamento teórico deste trabalho.

3 Gerência de Rede

Nos últimos anos o tamanho e a complexidade das redes de computadores vêm crescendo junto com a demanda de comunicação exigida por sistemas de computação. Sistemas *on-line* são uma realidade hoje em dia e que em muitos casos precisam de uma comunicação estável e, principalmente, segura. Em uma rede com milhares de componentes de comunicação, um defeito ou utilização incorreta de um desses componentes pode causar uma falha em todo o sistema de comunicação.

As principais funções da Gerência de Rede são manter estáveis condições de funcionamento, monitorá-las e controlar seus componentes de forma a maximizar seu desempenho.

Com o passar dos anos, os conceitos de gerência de rede começaram a ser aplicados também em componentes de software como, por exemplo, configuração de serviços como HTTP, DNS, FTP.

Neste capítulo é feito um breve estudo sobre o que é Gerência de Redes e quais suas principais aplicações. São também apresentados dois protocolos de Gerência de Rede utilizados atualmente; ainda, a aplicação dos conceitos de gerência de rede para a configuração de sistemas e, ao final serão apresentadas algumas considerações relevantes deste capítulo para o desenvolvimento deste trabalho.

3.1 Gerência de Redes e de Sistemas

Embora os conceitos de Gerência de Rede tenham sido desenvolvidos para componentes físicos, com o passar do tempo e de acordo com as necessidades de gerência seus conceitos foram aplicados também para a gerência de sistemas [12].

A gerência de redes e de sistemas é feita através de um conjunto de aplicações de gerência. Essas aplicações utilizam como base uma arquitetura de gerência formada

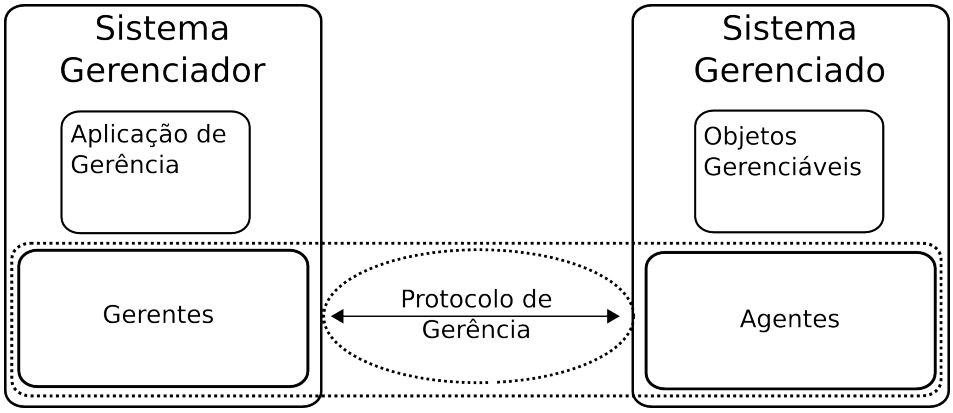

Figura 2 – Arquitetura de gerência de rede e de sistemas.

basicamente por gerentes, agentes e um protocolo de comunicação. Através dessa arquitetura é possível que aplicações realizem gerência de objetos em sistemas remotos. A Figura 2 ilustra os componentes envolvidos neste processo.

Essa arquitetura de gerência pode ser facilmente aplicada em diversos contextos, como por exemplo, para gerência de componentes de software. Neste caso, é preciso apenas implementar agentes específicos que atuem nos objetos gerenciáveis. A forma com que os gerentes manipulam as informações acontece da mesma maneira que na gerência de componentes de rede e pode variar dependendo do protocolo de gerência que se está utilizando.

Atualmente os protocolos SNMP (*Simple Network Management Protocol*) [13] e WBEM (*Web-Base Enterprise Management*) [14] são utilizados tanto para gerência de dispositivos quanto para a gerência de componentes de software. Na próxima seção será feita a apresentação desses dois protocolos.

3.2 Protocolos de Gerência

3.2.1 WBEM - *Web-Based Enterprise Management*

WBEM é um conjunto de tecnologias de gerência e de recursos comuns da Internet que tem como objetivo controlar recursos de ambientes distribuídos. O WBEM permite

a criação de componentes de gerência de forma rápida e objetiva facilitando a administração de diversos tipos de sistemas.

A arquitetura do sistema de gerência proposto pelo WBEM oferece a possibilidade de interação remota através do protocolo de comunicação HTTP (*HyperText Transfer Protocol*) , comum em comunicações via Internet. Através deste recurso pode-se definir um cliente (ou gerenciador) capaz de se comunicar com um sistema gerenciável através de uma rede simples com suporte a comunicação via HTTP. A Figura 3 ilustra a arquitetura do WBEM.

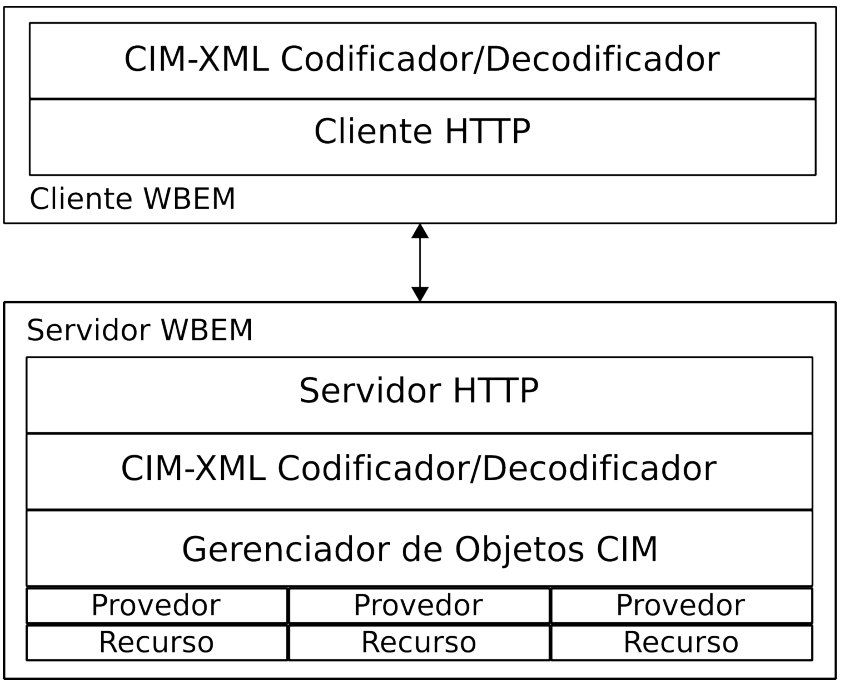

Figura 3 – Arquitetura do WBEM.

Do lado do cliente (ou gerente) o sistema recebe uma descrição CIM-XML [15], protocolo de comunicação utilizado pelo WBEM. Essa descrição é submetida ao servidor através do protocolo HTTP. O servidor interpreta a descrição e consulta a base mantida pelo Gerenciador de Objetos CIM (*Common Information Model*) [16] . Este componente é responsável por manter uma base de informações dos objetos gerenciáveis presentes no servidor. Essas informações podem ser lidas diretamente desta base ou obtidas em tempo de execução através dos provedores. Os provedores são responsáveis por ler e escrever informações nos recursos do servidor sejam estes softwares ou

hardwares.

Basicamente, o cliente pode executar três instruções com relação aos objetos hospedados no servidor (dependendo do nível de segurança): criar, modificar ou excluir um objeto. A única intervenção por parte do servidor ao cliente são os avisos de modificações no sistema.

A gerência remota oferecida pelo WBEM é implementada através de *web services* [17]. Este modelo de implementação possibilita a criação de sistemas *on-line* utilizando conceitos comuns da web como, a linguagem *XML* e o protocolo HTTP. Através desta tecnologia, a integração da gerência oferecida pelo WBEM com outros sistemas é facilitada e possibilita maior produtividade.

Embora o WBEM seja uma tendência no mercado de gerência de ambientes, ainda existem resistências com relação às tecnologias aplicadas no seu desenvolvimento. Na próxima seção é apresentado um protocolo de gerência de ampla utilização no mercado e ao final, será feito um breve comparativo entre os dois modelos de gerência.

3.2.2 SNMP - *Simple Network Management Protocol*

O SNMP foi implementado inicialmente para gerência de redes de computadores. No entanto, sua flexibilidade permitiu sua aplicação em diferentes ambientes, possibilitando a gerências de hardwares e softwares. A estrutura deste protocolo é baseada em gerentes e agentes, que se comunicam através de um acesso direto.

O objeto a ser gerenciado é descrito em uma base de informações que no SNMP é chamada de MIB (*Management Information Base*) [13]. Essa estrutura contém informações sobre os objetos e também sobre os agentes. A MIB tem a estrutura de árvore e pode ser dividida em sub-árvores, cada uma para um tipo de gerenciamento. Nessa estrutura, cada fabricante de produtos de rede possui uma ramificação descrevendo os objetos utilizados na gerência de seus produtos. A posição de um determinado objeto na árvore é determinada por uma seqüência de números, identificando os nós da árvore. A identificação, denominada OID (*Object IDentifier*) , é dividia por pontos e apresentada como `x.y.z`.

Um gerente SNMP consulta ou modifica um determinado objeto através de instruções de *get* e *set*, respectivamente. Junto a essas instruções é passado o *host* e o OID do

objeto. No caso da instrução *set* é também passado o tipo e o valor a ser modificado no objeto. Além disso, o gerente pode solicitar a um determinado *host* alertas de modificações de um objeto. Esse aviso é feito, no SNMP, ao gerente que o requisitou através da instrução chamada *trap*. A Figura 4 ilustra a comunicação entre o gerente e o objeto gerenciável.

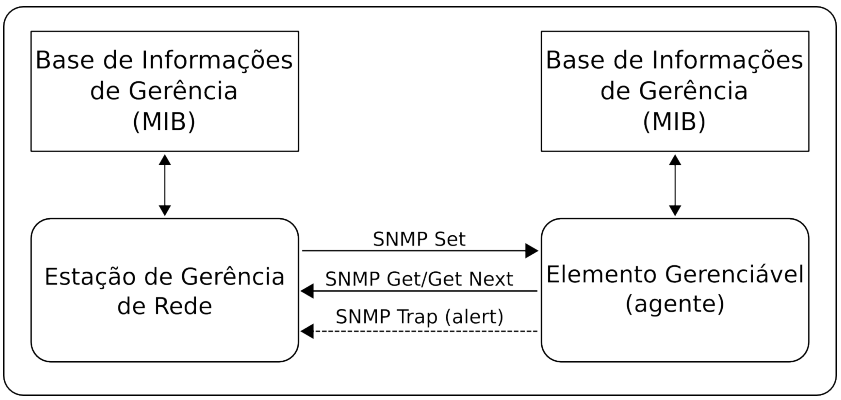

Figura 4 – Arquitetura do SNMP.

A flexibilidade do protocolo de gerência SNMP está na possibilidade de definição de novos objetos. Junto a estes objetos é preciso a implementação de um agente. Neste caso, o agente pode fazer gerencia tanto de elementos de hardware quanto de software. Outro ponto relevante na utilização do protocolo SNMP é que se podem criar estruturas com um gerente que controla diversos elementos, ou ainda, um mesmo elemento pode ser gerenciado por vários gerentes (*NxN*). A Figura 5 apresenta uma aplicação genérica do protocolo SNMP onde um gerente controla diferentes tipos de objetos.

Embora protocolos de gerência de rede como o WBEM proponham a utilização de tecnologias mais atuais de comunicação e integração com sistemas, atualmente, no mercado, o protocolo de gerência SNMP é encontrado com maior freqüência. Sua simplicidade de implementação facilita sua aplicação em sistemas embarcados o tornando um padrão em equipamentos como roteadores, *switches*, ADSL modems, entre outros.

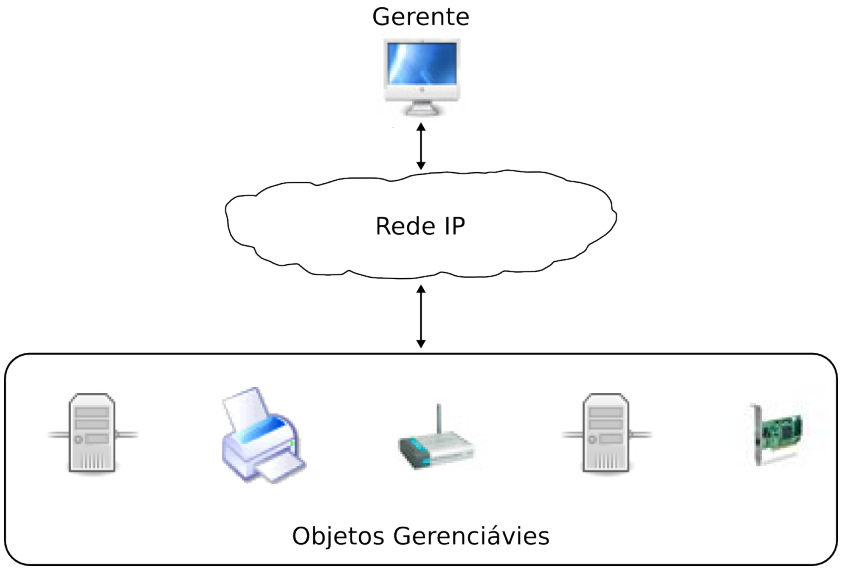

Figura 5 – Gerência de objetos com SNMP.

3.3 Considerações deste capítulo

É eminente a utilização de recursos computacionais em diversas áreas, tanto na indústria, quanto no comércio e na academia. A monitoração e controle desses recursos computacionais são feitas por protocolos de gerência como os apresentados neste capítulo. Os dois protocolos apresentados compartilham o mesmo objetivo: gerenciar recursos computacionais. No entanto, o que os diferencia é a forma de implementação que torna um ou outro melhor aplicável a determinados ambientes.

Máquinas Virtuais, suas interfaces de rede e demais recursos, podem ser gerenciados com o auxílio desses protocolos. Esses dois temas, virtualização e gerência de recursos, impulsionam o desenvolvimento deste trabalho que visa utilizar os recursos da virtualização de hardware para criação de ambientes virtuais. Tendo como principal foco os aspectos de rede dos ambientes, o próximo capítulo apresenta uma arquitetura para gerência desses recursos. No desenvolvimento deste trabalho o protocolo SNMP foi escolhido para a tarefa de gerência. Esta escolha foi feita a partir da comparação dos dois protocolos (WBEM e SNMP) de forma que o SNMP apresenta maior facilidade de integração com o trabalho e, também, pelo fato de ser atualmente um padrão para gerência em diversos tipos de sistemas.

4 Gerência de Rede de Máquinas Virtuais

Nos capítulos anteriores apresentaram-se as tecnologias utilizadas neste trabalho, que tem como contexto a proposta de utilizar a virtualização de hardware para criação de ambientes virtuais. Essa proposta surge como uma alternativa no intuito de melhor utilizar os recursos computacionais e prover um ambiente controlado para, por exemplo, criação de ambientes para avaliação de aplicações distribuídas.

Neste capítulo será apresentada uma arquitetura para configuração e gerência dos recursos de rede de ambientes virtualizados. Questões como mapeamento de conexões, o método de gerência utilizado para configuração dos recursos de rede e as ferramentas utilizadas na intenção de aproximar o comportamento de rede dos ambientes reais e virtuais serão apresentados e discutidos.

No primeiro momento do capítulo discutem-se os elementos da arquitetura de gerência da rede de máquinas virtuais, organizada em três níveis. Logo após são apresentados os componentes envolvidos na comunicação de rede em máquinas virtuais Xen [18]. Essas duas seções servirão como base de integração da arquitetura de gerência de rede para máquinas virtuais baseadas em Xen. Ainda, será feita uma análise sobre a integração da arquitetura com outros sistemas de virtualização.

4.1 Arquitetura de Gerência de Rede de Máquinas Virtuais

Com o objetivo de configurar a rede de um ambiente de máquinas virtuais, desenvolveu-se uma arquitetura de gerência capaz automatizar este processo. Esta arquitetura foi dividida em três níveis, de forma a facilitar a apresentação de seus componentes. A Figura 6 ilustra a interação entre estes níveis, de forma que é possível identificar, no nível um, dois componentes de descrição utilizados como entrada da arquitetura. Abaixo, no nível dois, encontra-se o módulo de gerência de rede, e no nível mais abaixo

na figura as máquinas do ambiente. Essas máquinas devem ser previamente instaladas com um sistema virtualizador e com suporte a gerência de recursos via SNMP.

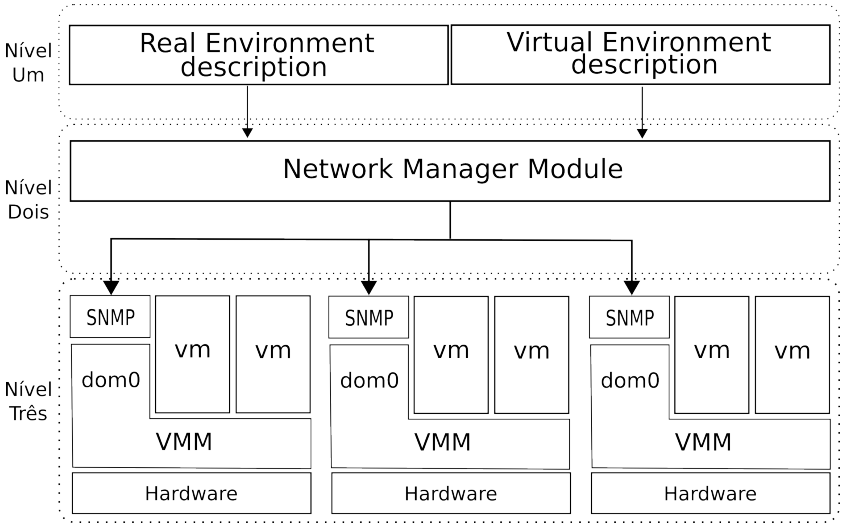

Figura 6 – Arquitetura do Modelo de Gerência de Rede de Máquinas Virtuais.

a) Componentes de Entrada da Arquitetura

Os dois componentes de descrição, representados no nível um da figura, são arquivos *XML* (*eXtensible Markup Language*). O primeiro descreve o ambiente real apresentando informações como características de rede e conexões entre as máquinas físicas e quais máquinas virtuais cada uma hospeda. O segundo arquivo, de descrição do ambiente virtual, apresenta a estrutura de rede que se deseja configurar entre as máquinas virtuais.

A estrutura do arquivo que descreve o ambiente físico (ou real) é organizada de forma que se possa representar conjuntos de máquinas reais divididas em grupos, chamados de *SubNets*. As máquinas destes grupos são conectadas entre si e a um determinado *Router* por meio de um *NetworkLink*. Outras informações relevantes para gerência de rede, a cerca das máquinas virtuais, também são informados nesse arquivo: número de interfaces de rede e como elas são identificadas pelo sistema de virtualização.

A Figura 7 ilustra a estrutura do arquivo *XML* referente a descrição do ambiente real. No Apêndice A é apresentado o modelo DTD (*Document Type Definitions*) [19]

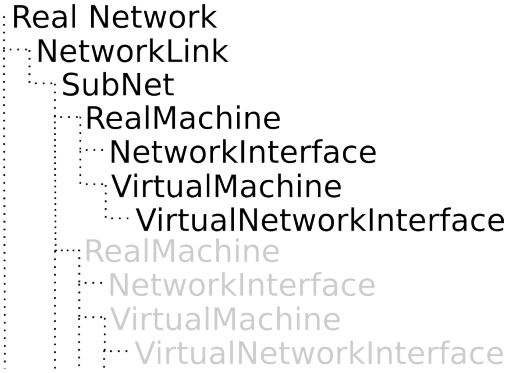

Figura 7 – Estrutura do Arquivo *XML* para descrição do ambiente real.

de descrição do arquivo junto com um exemplo de construção nesse formato.

O arquivo que descreve o ambiente virtual é organizado de forma que se possa representar a estrutura de uma rede de computadores, neste caso formada por máquinas virtuais. Nele são descritas *SubNets* de máquinas, conectadas a um *Router* por um - *NetworkLink* – que, por sua vez, possui informações referentes a vazão e a latência com que a *SubNet* irá se comunicar com o seu referente *Router*. Cada *SubNet* descrita no arquivo pode ter N máquinas que são identificadas pelo nome. Além disso, cada uma das máquinas pode ter N interfaces de rede que são identificadas pelo endereço *MAC*. O nome da máquina e o endereço *MAC* das interfaces são utilizados para cruzar informações com o arquivo de descrição do ambiente real. O cruzamento dessas informações completa a seqüencia de dados utilizados para criação do ambiente virtual.

A Figura 8 ilustra a estrutura do arquivo *XML* referente a descrição do ambiente virtual que se deseja configurar. No Apêndice A é apresentado o modelo DTD (*Document Type Definitions*) de descrição do arquivo junto com um exemplo de construção de uma descrição neste formato.

Por fim, esses dois arquivos são utilizados como entrada para a camada intermediária da arquitetura de gerência, o Módulo de Gerência de Rede, ilustrado na Figura 6.

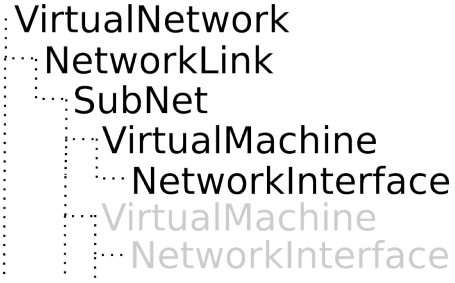

Figura 8 – Estrutura do Arquivo *XML* para descrição do ambiente virtual.

b) Módulo de Gerência de Rede

O Módulo de Gerência de Rede é uma aplicação desenvolvida em Java [2] que tem por principal objetivo mapear as informações descritas nos arquivos de entrada e configurar um ambiente baseado em máquinas virtuais.

O primeiro procedimento deste nível da arquitetura é carregar os arquivos de configuração e validar os dados de acordo com o arquivo de descrição de tipo de documento (DTD). No Apêndice A foram descritos dois exemplos de arquivos *XML*, um com a descrição de um ambiente real e outro com a descrição de um ambiente virtual.

O Módulo de Gerência de Rede é um conjunto de classes estruturadas de forma a facilitar a configuração do ambiente real e, assim, criar o ambiente virtual desejado. A Figura 9 ilustra a relação de dependência entre as classes da aplicação. Esta relação de dependência foi inspirada na estrutura dos arquivos de descrição.

O primeiro arquivo lido é o de descrição do ambiente real. O módulo cria o objeto da classe *RealNetwork* e faz a leitura dos elementos do próximo nível do arquivo de descrição. Neste caso, o módulo cria o objeto da classe *NetworkLink* e o armazena em uma lista como atributo da classe *RealNetwork* relacionada. Esse procedimento se repete para todos os tipos de classes e para todos os níveis da estrutura descrita no arquivo, até que todos os elementos e suas informações sejam carregados na estrutura de objetos do módulo.

No momento seguinte é feita a leitura do arquivo de descrição do ambiente virtual. O procedimento é basicamente o mesmo que o aplicado à leitura da descrição do ambiente real: o módulo cria o objeto da classe *VirtualNetwork*; os objetos da classe *Networ-*

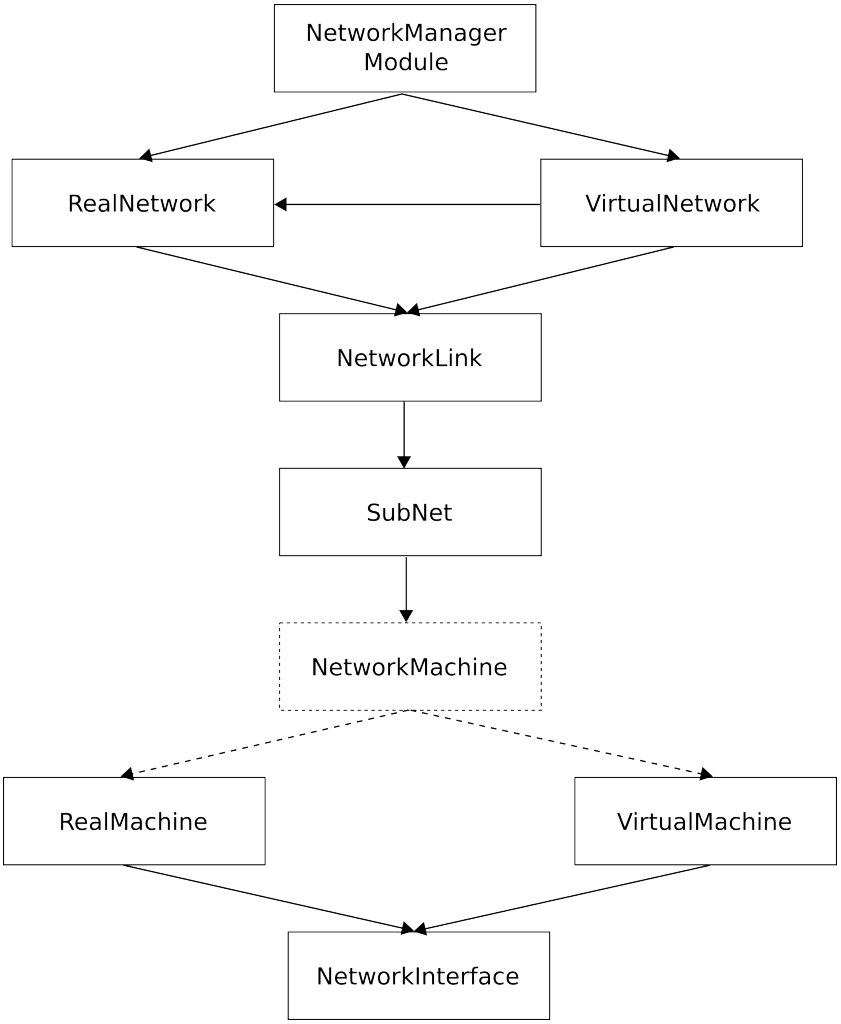

Figura 9 – Estrutura de classes do Módulo de Gerência de Rede.

kLink, armazenando como atributo da classe *VirtualNetwork* relacionada; e assim sucessivamente. A diferença nesta leitura está na organização dos objetos relacionados às máquinas virtuais, que não são criadas e, sim, relacionadas à aquelas criadas pela leitura do ambiente real. As máquinas virtuais são identificadas através do nome e do endereço *MAC* para que se complete as informações de rede referentes ao ambiente virtual que se deseja criar.

Com isso, o Módulo de Gerência de Rede de Máquinas Virtuais tem a visão da estrutura do ambiente de dois diferentes pontos: do ambiente real, com informações de máquinas e redes reais; e do ambiente virtual que se deseja criar com as máquinas

virtuais.

A partir dessas informações o módulo gera um *script* de configuração, que é armazenado em cada um dos objetos que representam as máquinas reais do ambiente, *RealMachine*. Esse *script* possui instruções referentes a configurações de rede, como regras de tráfego e controle de vazão e latência entre as interfaces de rede das máquinas virtuais. No entanto, a construção desse *script* – utilizado para configurar o Monitor de Máquinas Virtuais - depende das ferramentas de controle de rede disponíveis. Já as informações utilizadas para a configurações das máquinas virtuais são extraídas do arquivo de definição do ambiente virtual.

Após os *scripts* de configuração das máquinas reais e as informações das máquinas virtuais estarem completas é possível fazer a configuração do ambiente. Neste trabalho utilizam-se os conceitos de gerência de rede baseados no protocolo SNMP, apresentado no Capítulo 3, Seção 3.2.2.

Em sistemas paravirtualizados, uma máquina virtual chamada de domínio privilegiado, identificado na Figura 6 como *dom0*, tem acesso a todos os recursos de hardware e de softwares presentes, como apresentado no Capítulo 2. Sendo assim, a arquitetura apresentada neste trabalho interage apenas com o serviço SNMP presente neste domínio privilegiado para configurar o ambiente. Na Figura 6 pode-se perceber essa interação entre os dois níveis inferiores da arquitetura.

Para configurar as máquinas virtuais e seus componentes de rede através do SNMP uma MIB (*Management Information Base*) com suporte a gerência dos recursos virtuais deve ser instalada. A MIB utilizada na implementação deste trabalho foi desenvolvida em paralelo, dentro do grupo de pesquisa ao qual este trabalho também faz parte. [20] apresenta a *vMIB*, que descreve os componentes físicos e virtuais de um sistema paravirtualizado que podem ser monitorados e/ou modificados.

Os componentes das máquinas virtuais gerenciados pela *vMIB* utilizados neste trabalho são as interfaces de rede das máquinas virtuais. Através deste recurso pôde-se fazer o controle de atributos como endereço de rede, definições de máscara para formação de sub-redes de máquinas, definições de rotas de rede, dentre outras configurações pertinentes à comunicação entre máquinas virtuais. Assim, o módulo de gerência modifica as informações dos objetos referentes as interfaces de rede das máquinas virtuais de forma direta, ou seja, mapeia a configuração tal qual foi lido do arquivo de descrição do ambiente virtual.

Após a configuração das interfaces das máquinas virtuais a rede do ambiente virtual está parcialmente configurada, porém, ainda faltam definições como controle de tráfego, vazão e latência. Essas configurações podem ser feitas de diferentes maneiras, dependendo do sistema paravirtualizador utilizado. Para o módulo de gerência a diferença está na geração dos *scripts* de configuração, que são submetidos às máquinas reais do ambiente.

Na próxima seção apresentam-se as ferramentas e os componentes de rede no sistema de paravirtualização Xen. Após, apresenta-se a integração do Módulo de Gerência de Rede com este sistema de paravirtulização.

4.2 Comunicação de Rede entre Máquinas Virtuais Xen

O Monitor de Máquinas Virtuais Xen [18] é um projeto desenvolvido pela Universidade de Cambridge [21], amplamente utilizado no meio acadêmico por ser distribuído livremente.

Seguindo o modelo de virtualização de hardware, apresentado na Seção 2.2, o Xen é uma camada de software localizada entre o hardware e o sistema operacional. Na inicializaçao do sistema uma máquina virtual com acesso privilegiado aos componentes de software e hardware, chamada de *dom0*, é criada para oferecer suporte a inicialização de outras máquinas virtuais, com menor privilégio de execução, chamadas *domU*s.

Cada *domU* possui recursos virtuais de hardware como memória, CPU, disco e rede. Esses componentes são disponibilizados isoladamente a cada *domU* para que se possa executar um sistema operacional na máquinas virtuais de forma independente. Ou seja, o sistema operacional executa como se em uma máquina sem virtualização e sem interferir na execução das outras máquinas virtuais.

O suporte a comunicação de rede oferecida às máquinas virtuais pelo Xen é composta de um conjunto de elementos reais e virtuais. As interfaces de rede virtuais, responsáveis pela comunicação das máquinas virtuais, são componentes de software, implementados pelo Xen, formados por dois *buffers* – um para envio e outro para recebimento de pacotes de rede. Os *buffers* enfileiram os pacotes de rede que são enviados para a rede, no caso do *buffer* de envio, ou enviados para a aplicação da determinada

máquina virtual, no caso do *buffer* de recebimento.

O *dom0* implementa também um componentes de software chamado *switch* virtual, que é encarregado de encaminhar pacotes entre as interfaces de rede virtuais. Neste *switch* também estão conectadas as interfaces de rede reais, possibilitando a comunicação remota das máquinas virtuais hospedadas em um determinado *host*.

A função do *switch* é encaminhar os pacotes sob demanda às outras interfaces virtuais, caso a comunicação seja entre máquinas virtuais; ou encaminhar os pacotes à interface de rede real. Este encaminhamento segue um conjunto de regras, especificadas pelo *iptables* [22], no *dom0*.

Esses componentes garantem a comunicação de rede entre máquinas virtuais localizadas em um mesmo computador ou em computadores remotos. A Figura 10 ilustra os componentes envolvidos na comunicação de rede do Xen.

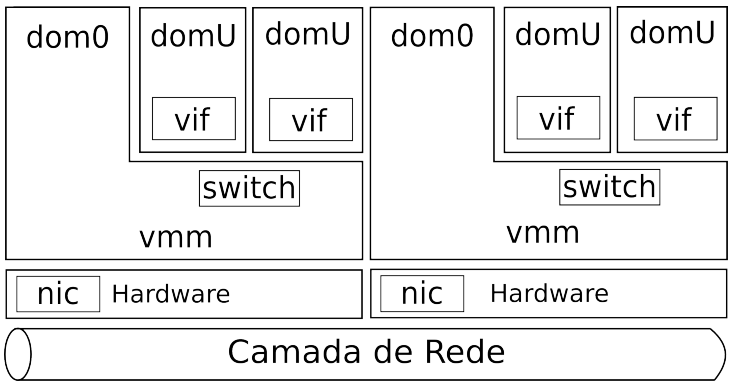

Figura 10 – Arquitetura de Rede do Monitor de Máquinas Virtuais Xen.

4.3 Gerência de Rede de Máquinas Virtuais Xen

Com base nos conhecimento da estrutura de comunicação do Xen pode-se, então, configurar os elementos responsáveis por definir regras de comunicação como isolamento, vazão e latência de rede entre máquinas virtuais. Como mencionado na Seção 4.1, o nível inferior ao Módulo de Gerência de Rede na arquitetura proposta refere-se ao sistema paravirtualizador e às máquinas utilizadas para criação do ambiente de emu-

lação de sistemas distribuídos. Nesta seção será apresentada a integração do Módulo de Gerência de Rede com o sistema de paravirtualização Xen, instalado em um conjunto de máquinas conectadas por uma rede.

Seguindo a arquitetura proposta neste capítulo, após as etapas de leitura dos arquivos de configuração e do mapeamento das informações referentes as máquinas virtuais é feita então a configuração dos domínios privilegiados de cada uma das máquinas reais, no Xen chamados de *dom0*.

Este processo é feito com a submissão dos *scripts* de configuração gerados nos objetos do módulo referentes a cada uma das máquinas reais do ambiente, através de um agente SNMP, que recebe o *script* e executa as instruções no *dom0* do *host*.

O *script* de configuração é dividido em três partes: configuração de regras de tráfego, definição de vazão de rede e definição de latência de rede entre as máquinas virtuais. Ambos os três controles são feitos na camada de rede da pilha do protocolo TCP/IP [23]. Além disso, as configurações são replicadas em todas as máquinas reais, garantindo os requisitos de isolamento e controle de tráfegos feitos no arquivo de descrição do ambiente virtual, independentemente da localização das máquinas virtuais. Ou seja, embora máquinas de sub-redes diferentes estejam hospedadas em uma mesma máquina real, o controle feito no domínio privilegiado garantirá que essas máquinas não se comuniquem através da rede.

Para exemplificar melhor esta situação supõem-se uma máquina real que hospeda duas máquinas virtuais de sub-redes diferentes, onde o objetivo do controle é o isolamento de rede entre as máquinas. Para este caso, defini-se então duas regras aplicadas na tabela de encaminhamento de pacotes do *iptables* do domínio privilegiado da máquina, como as apresentadas na Tabela 2. A primeira regra veta o encaminhamento de pacotes que tem como origem a rede $192.168.1.0/24$ e como destino a rede $192.168.2.0/24$. A segunda regra veta o encaminhamento de pacotes no sentido inverso.

Tabela 2 – Regras de isolamento de rede no *iptables*.

target	prot	opt	source	destination
DROP	tcp	–	192.168.1.0/24	192.168.2.0/24
DROP	tcp	–	192.168.2.0/24	192.168.1.0/24

As definições de vazão e latência de rede também são configuradas de forma a limitar a comunicação entre sub-redes de máquinas. Assim, através do comando `tc` (*traffic*

control) [24] do Linux utilizam-se algoritmos de enfileiramento e atraso de pacotes para simular o controle de vazão e latência da rede.

Para exemplificar como são feitas essas configurações, supõem-se o mesmo ambiente proposto para o isolamento de rede apresentado anteriormente. Após o isolamento, feito utilizando as regras do *iptables*, deseja-se definir uma vazão de $5MBit/s$ e $20ms$ de latência entre as duas sub-redes.

Para a implementação deste problema, o comando *tc* cria um *buffer* onde são armazenados os pacotes que serão liberados na rede, novamente, de acordo com um tempo de espera (atraso ou latência). Este controle é feito pelo módulo NetEm [25].

A simulação de vazão é feita pelo módulo TBF [26] e é também controlado através do comando *tc*. O módulo intercepta os pacotes e adiciona, no final, um "conteúdo virtual", O que irá definir a vazão com que os pacotes serão transmitidos. Este conteúdo é nulo e é descartado pelo sistema que irá receber o pacote. O calculo do tamanho deste "conteúdo virtual" é feito pelo próprio modulo que implementa este controle, de forma que o usuário apenas informa a capacidade total do link e a vazão que se deseja limitar.

Ao final da replicação dessas configurações em todas as máquinas reais, a rede do ambiente baseado em máquinas virtuais está pronta para comunicar de acordo com as configurações passadas nos arquivos de descrição do ambiente.

4.4 Portabilidade da Arquitetura de Gerência de Rede

A arquitetura de gerência de rede proposta neste capítulo foi modelada para suportar sua implementação também em outros sistemas de virtualização, que não o Xen. Em sistemas baseados em Linux, e que seguem o modelo de paravirtualização – onde uma máquina virtual privilegiada gerencia as demais máquinas virtuais, a arquitetura poderá se adaptar com a modificação de pequenas funções, como detalhes referentes aos aspectos de rede entre os sistemas de virtualização. Isso porque os aspectos de isolamento de rede e controle de vazão e latência são implementados por ferramentas comuns do Linux, portanto não precisam ser modificadas.

A principal mudança estaria na necessidade do desenvolvimento de uma agente SNMP para gerência dos recursos de rede das máquinas virtuais. Dentre os sistemas

de virtualização baseados em Linux pode-se citar kvm [27], OpenVZ [8], Virtuozzo [9] e User-Mode Linux [28]. Por outro lado sistemas de virtualização não baseados em Linux precisariam de adaptação de toda a gerência dos recursos de rede para controle de vazão, latência e isolamento. Sistemas como VMWare [29] e Microsoft Virtual Server [30] também oferecem características de virtualização – como isolamento de VMs e compartilhamento de recursos – porém possuem ferramentas de controle de rede pertinentes ao sistemas operacional ou, em alguns casos, específicas, como a ferramenta de gerência centralizada para o VMWare [31].

4.5 Considerações deste capítulo

Este capítulo apresentou uma proposta de arquitetura de configuração de rede para a criação de ambientes distribuídos baseados em máquinas virtuais. A arquitetura tem como entrada dois arquivos *XML*, de descrição do ambiente real, e das definições de rede do ambiente virtual que se deseja criar. O Módulo de Gerência de Rede faz a leitura desses arquivos e mapeia a configuração no ambiente através de um protocolo de gerência de rede. Ao final do processo o conjunto de máquinas virtuais deverá se comportar assim como um ambiente de máquinas reais conectadas entre si.

Para avaliar a arquitetura e a efetividade da configuração proposta, um *framework* de criação de ambientes distribuídos utiliza os recursos da arquitetura para configurar a rede de um ambiente. Nele são feitos testes, através da execução de uma aplicação distribuída real, na intenção de avaliar a configuração proposta pela arquitetura. A especificação e os resultados dos testes são apresentados no próximo capítulo.

5 Avaliação e Resultados

Este capítulo apresenta a avaliação da arquitetura de gerência de rede de máquinas virtuais em dois momentos. No primeiro, serão feitos testes para avaliar a efetividade de controle de vazão e latência de rede, feitos pelas ferramentas apresentadas na seção 4.3, em máquinas com o sistema de virtualização Xen. Após, a arquitetura será aplicada em um contexto onde, além da gerência de rede, são feitos procedimento de criação e configuração de ambientes distribuídos para avaliação de aplicações. Neste contexto, a arquitetura é utilizada para configurar as conexões de rede de um ambiente de *Grid* na intenção de isolar conjuntos de máquinas virtuais e simular atrasos na comunicação entre elas.

5.1 Efetividade do Controle de Rede de Máquinas Virtuais

Para demonstrar a efetividade do controle de largura de banda e a perda de desempenho que as ferramentas causam na comunicação entre máquinas virtuais, criou-se um conjunto de testes, a partir dos quais se conseguiu obter resultados para avaliação da abordagem.

As máquinas reais utilizadas para os testes são cada uma um Pentium 4 2.8GHz com 1MB de cache e 2.5GB de RAM. As máquinas são conectadas entre si por uma rede *Fast Ethernet*, tem Xen VMM 3.1 e o *dom0* utiliza 328MB da memória disponível pela máquina. Em ambos os testes, de vazão e latência, a configuração foi feita nas interfaces virtuais, localizadas no nível privilegiado da virtualização de hardware, relacionadas às interfaces de suas respectivas máquinas virtuais.

Nos testes, o controle de largura de banda foi feito com a ferramenta TBF (*Token Bucket Filter*) usando o comando *tc* (*Linux Traffic Control*). Um arquivo de 300MB foi transferido através do protocolo SCP (*Secure Copy Protocol*) [32] em dois cenários:

entre máquinas virtuais hospedadas uma mesma máquina real (comunicação local), e entre máquinas virtuais hospedadas diferentes máquinas reais (comunicação remota).

Na Figura 11 é apresentado o gráfico plotado com os resultados da transmissão do arquivo em diferentes larguras de banda, configuradas com as ferramentas apresentadas na seção 4.3. Foram criados dez casos onde a largura de banda é acrescida em 100% em relação ao caso anterior, começando em 256KBit/s até 10MBit/s. A diagonal principal demonstra o valor de banda configurado e as linhas pontilhadas apresentam os valores obtidos na transferência do arquivo para ambos os testes – comunicação local e remota. A tabela 3 apresenta as médias obtidas na transferência do arquivo junto com o desvio padrão através de 32 amostras coletadas durante a transmissão do arquivo.

Tabela 3 – Médias das vazões obtidas em 32 amostras.

Vazão configurada	Local	Remoto	Desvio Padrão
32KB/s	29.77	29.77	0.74
64KB/s	59.77	59.75	1.18
128KB/s	119.78	119.76	1.92
256KB/s	239.54	239.51	3.10
512KB/s	489.64	486.91	13.54
1024KB/s	918.55	918.55	22.86
2048KB/s	1909.90	1909.90	47.91
4096KB/s	3739.90	3739.90	92.86
8192KB/s	6909.90	6909.90	172.04
12500KB/s	10219.90	10219.90	255.45

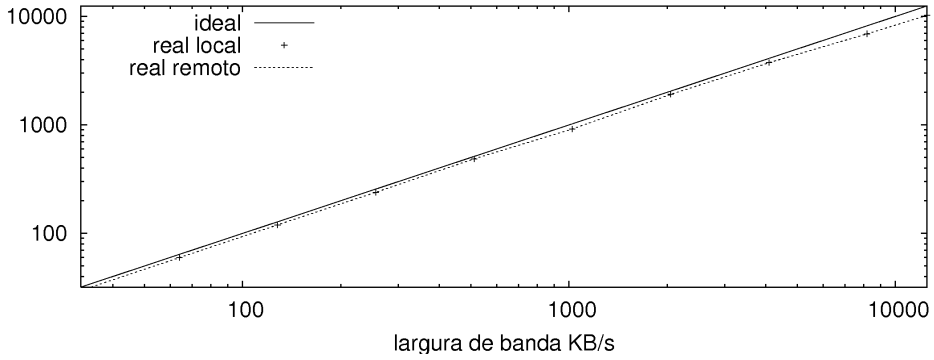

Figura 11 – Largura de banda de rede limitada entre Máquinas Virtuais Locais e Remotas.

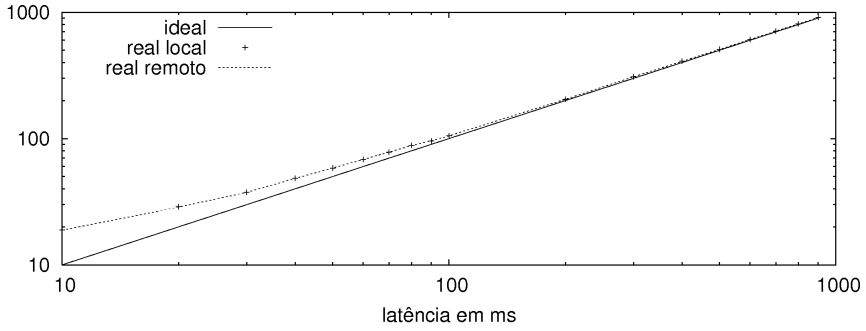

Figura 12 – Latência de rede limitada entre Máquinas Virtuais Locais e Remotas.

A análise do gráfico permite-nos concluir que a perda de desempenho na comunicação de rede acentua conforme a largura de banda vai aumentando. Em um ambiente distribuído, menores larguras de banda são mais comuns visto que, hoje em dia, o meio mais comum de comunicação entre sites remotos é a Internet, não havendo então uma rede exclusiva para a execução de aplicações. Neste caso, os experimentos demonstram valores satisfatórios para a proposta de utilizar ferramentas de controle de rede e virtualização de hardware na emulação de sistemas distribuídos.

De uma mesma maneira, foram feitos testes para avaliar a latência na rede entre os dois casos (comunicação local e remota). A configuração deste recurso foi feita utilizando a ferramenta NetEm (Network Emulator) com o comando *tc* (*Linux Traffic Control*). Para mensurar os tempos de latência foram transferidos pacotes ICMP [33] nos dois cenários (máquinas virtuais locais e remotas). Foram criados dez casos onde a latência foi aumentada em $10ms$ com relação ao caso anterior, variando entre $10ms$ até $100ms$. A partir desse valor a latência foi aumentada em $100ms$ criando-se mais oito casos, até $900ms$. A tabela 4 apresenta a médias dos valores obtidos em 1000 amostras feitas durante a transmissão dos pacotes ICMP.

A Figura 12 apresenta o gráfico plotado com os tempos de resposta na transmissão dos pacotes entre os dois cenários. A linha cheia apresenta o tempo de latência configurado e as linhas pontilhadas apresentam os tempos de latência obtidos. Foi possível constatar que quanto menor o valor de latência configurado, maior é a diferença entre o esperado e o obtido.

Em ambientes distribuídos, assim como mencionado com relação a largura de banda, os valores de latência são normalmente altos. Neste caso, utilizando a ferramenta de

Tabela 4 – Médias das latências obtidas em 1000 amostras.

Latência configurada	Local	Remoto	Desvio Padrão
10ms	18.95	18.95	1.19
20ms	28.76	28.70	1.31
30ms	37.49	37.51	2.93
40ms	48.35	48.35	1.74
50ms	58.44	58.45	2.06
60ms	68.24	68.24	2.01
70ms	78.13	78.13	2.33
80ms	88.12	88.10	2.11
90ms	96.01	96.00	3.22
100ms	105.38	105.38	3.18
200ms	204.97	204.97	3.12
300ms	308.70	308.70	2.28
400ms	408.76	408.76	0.50
500ms	508.84	508.85	0.83
600ms	609.11	609.11	1.69
700ms	708.75	708.75	2.49
800ms	809.68	809.67	1.90
900ms	908.82	908.83	4.09

controle de latência apresentada no ambiente proposto, é possível emular ambientes com comportamentos mais próximos ao de ambientes reais, o que tornam satisfatórios os resultados obtidos nos testes.

5.2 Ambiente de Emulação de Sistemas Distribuídos baseado em Virtualização

A proposta de criar um *framework* para emulação de sistemas distribuídos baseado em virtualização é uma alternativa que tem como objetivos controlar a execução de uma aplicação neste ambiente, melhor utilizar os recursos computacionais disponíveis e integrar ferramentas *open-source* tanto para virtualização dos recursos físicos quanto para gerência do ambiente.

Na intenção de prover uma ambiente direcionado para usuários, o *framework*, que

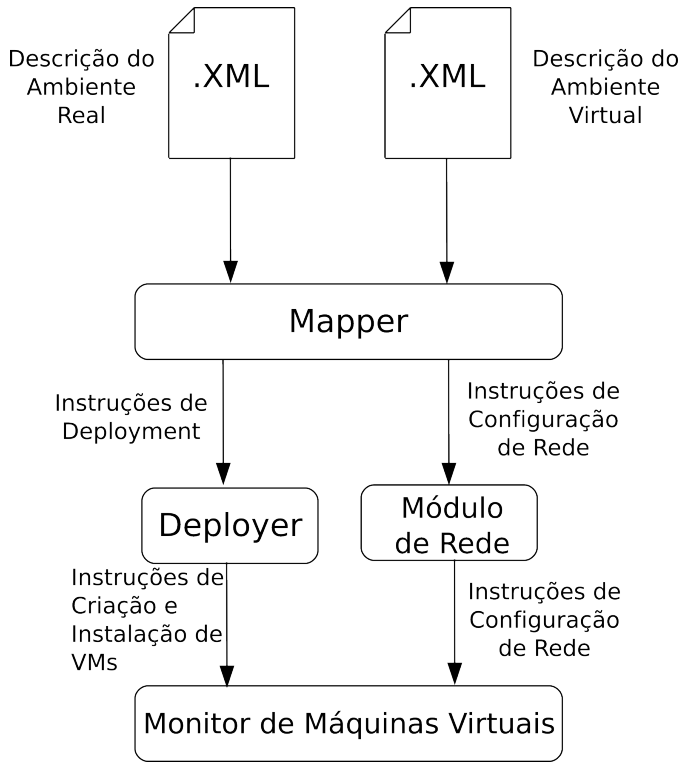

Figura 13 – Arquitetura do ambiente proposto.

será apresentado nesta seção, implementa um conjunto de componentes responsável por tarefas como instalar um ambiente virtual, controlar a execução de experimentos, gerenciar as conexões de rede e instalar serviços nos nós do ambiente. Após a execução de tarefas como instalação de serviços nas máquinas virtuais e configuração de vazão e latência de rede, o ambiente está pronto para a execução de aplicações como *Grid Computing*, teste de protocolos como P2P e aplicações de larga escala. A Figura 13 ilustra os componentes envolvidos e o fluxo de dados entre eles.

Dois arquivo *XML* (*eXtensible Markup Language*) são utilizado como entradas que descrevem respectivamente o ambiente real onde o sistema será instalado e o ambiente virtual que se deseja criar.

O arquivo que descreve o ambiente real possui informações referentes às características físicas das máquinas (memória, disco, CPU, rede) disponíveis para criação do ambiente e as conexões de rede entre essas máquinas (*switches*, roteadores). O segundo arquivo, que descreve o ambiente virtual que se deseja criar, possui informações como

o número de máquinas do ambiente, recursos (CPU, memória, disco) de cada máquina do ambiente e também a descrição da rede que conecta essas máquinas.

Em um primeiro momento o módulo de mapeamento do sistema interpreta os arquivos e gera duas saídas: uma encaminhada ao módulo de *deployment*, que utiliza ferramentas como [34], e outra encaminhada para o módulo de gerência da rede do ambiente.

O módulo de *deployment* é responsável por criar as máquinas virtuais nas suas respectivas máquinas reais, instalando os serviços requisitados pelo módulo de mapeamento. Dentre os serviços que podem ser instalados nas máquinas virtuais podemos citar servidores de banco de dados, clientes de uma rede P2P ou softwares para execução de tarefas de um *Grid*.

Por outro lado, o módulo de gerência de rede é responsável por configurar as conexões entre as máquinas virtuais tal qual a configuração definida no arquivo onde foi descrito o ambiente virtual. Questões como isolamento de redes, controle de trafego (vazão e latência) e definição de regras de comunicação são configuradas por este módulo com o objetivo de aproximar o comportamento do ambiente virtual ao de um ambiente real.

Ao final do processo de *deployment* e da configuração da rede, o ambiente está pronto para a execução da aplicação distribuída para o qual foi configurado. Neste momento é preciso a intervenção do módulo de gerência de experimentos, que é responsável por executar a aplicação de acordo com regras que visam explorar seus pontos de falha como, por exemplo, a disponibilidade de nós no ambiente ou mesmo a qualidade da conexão de rede entre um conjunto de máquinas. O gerenciador de experimentos analisa a execução da aplicação e gera relatórios que o usuário poderá analisar a fim de avaliar os problemas da aplicação.

5.3 Configuração e Controle de Rede em um Ambiente Distribuído Emulado

Utilizando o *framework* apresentado na seção anterior e a arquitetura apresentada no Capítulo 4 como seu módulo de configuração de rede, criou-se um ambiente de *Grid* e

executou-se uma aplicação de teste na intenção de avaliar as conexões de rede do ambiente. Os arquivos de descrição, utilizados como entrada da arquitetura de configuração de rede, dos ambientes reais e virtuais apresentados a seguir, econtram-se no Apêndice A.

5.3.1 O ambiente Físico

O ambiente físico utilizado para ciração do ambiente virtual utilizado na avaliação deste trabalho consiste em um COW (*Cluster of Workstations*). As máquinas reais do ambiente têm a mesma configuração das utilizadas no controle de vazão e latência apresentados na seção 5.1. As imagens das máquinas virtuais convidadas estão hospedadas no disco de seus respectivos hospedeiros. Cada uma das máquinas reais hospeda 8 máquinas virtuais, cada uma delas utilizando 256MB de memória RAM e compartilhando os demais recursos. No momento dos testes, somente o tráfego gerado pelos experimentos era presente da rede.

5.3.2 O ambiente Virtual

O ambiente virtual criado para implantação do sistema distribuído escolhido consiste em três conjuntos de máquinas virtuais. Cada um dos conjuntos pertence a uma sub-rede ao qual somente uma máquina virtual, o Proxy, tem permissão de comunicar-se com os *Proxies* dos outros conjuntos de máquinas. Cada um dos sites possui um número diferente de máquinas virtuais e máquinas virtuais de um mesmo site podem estar em diferentes máquinas reais, como mostra a Tabela 5 – onde cada linha corresponde a uma máquina real e as colunas correspondem aos conjuntos de máquinas virtuais. Assim podemos identificar em cada célula da tabela as máquinas correspondentes ao determinado site e hospedadas no determinado *host*.

Como se pode ver na Tabela 5, cada uma das máquinas reais hospeda 8 máquinas virtuais que podem pertencer a diferentes sites. No entanto, embora as máquinas virtuais estejam hospedadas na mesma máquina real, não existe comunicação direta entre elas. Isso porque, em um primeiro momento, elas estão isoladas pela rede através da

Tabela 5 – Distribuição das máquinas virtuais entre as máquinas reais e os sites.

	site virtual 1	site virtual 2	site virtual 3	total
host 1	vmgrid108, vmgrid109	vmgrid201-vmgrid205, vmgridpeer2		8
host 2		vmgrid206-vmgrid212	vmgrid308	8
host 3	vmgrid101-vmgrid107, vmgridpeer1			8
host 4			vmgrid301-vmgrid307, vmgridpeer3	8
total	10	13	9	32

definição de endereço IP e máscara de rede. Por outro lado, a tecnologia dos Monitores de Máquinas Virtuais oferece isolamento entre máquinas virtuais hospedadas em um mesmo *host*. Isso demonstra a possibilidade de criar domínios de redes isolados. As máquinas virtuais nomeadas `vmgridpeerX` em cada site funcionam como o *Proxy* de rede para comunicação entre as sub-redes. Cada máquina virtual pertencente a um determinado site tem acesso às outras máquinas deste mesmo site, mesmo que hospedadas em máquinas reais diferentes.

5.3.3 O ambiente de Grade

O *middleware* de grade utilizado neste trabalho foi o OurGrid [35]. No sistema de grade OurGrid, cada `peer` é responsável pela comunicação com outros `peers` no objetivo de conseguir recursos computacionais necessários para sua demanda local e também oferecer seus recursos, em um modelo econômico conhecido como o *network of favors* [35]. O escalonador MyGrid oferece recursos da grade para os usuários do Grid que executam o software chamado UserAgent, também chamado de SWAN.

Como apresentado na Firgura 14, cada site possui uma máquina virtual que tem o papel de `peer`. Esta máquina funciona também como *Proxy* de rede e está apta a

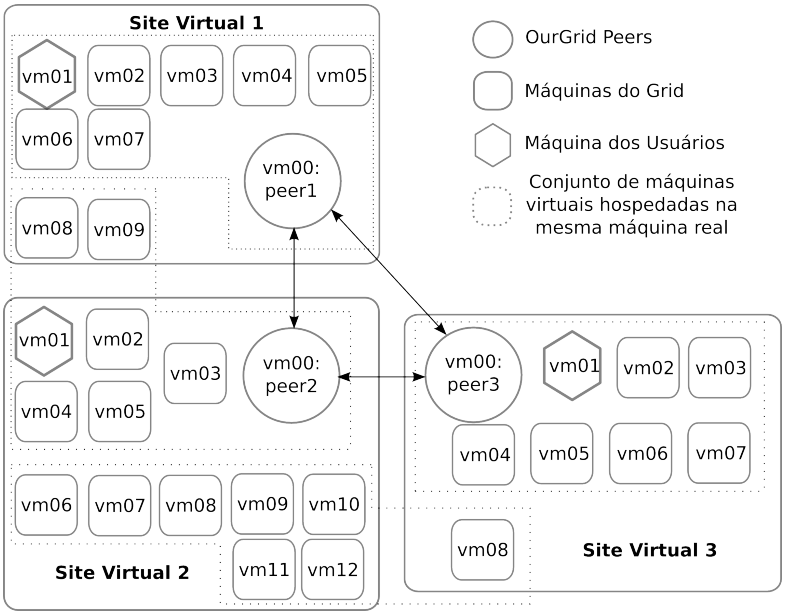

Figura 14 – Ambiente de Grid Virtuais criado para os testes.

comunicar-se com os outros `peers`. Estes conhecem uns aos outros através do componente `corepeer`, (não apresentado na figura), instalado no site virtual 1 na mesma máquina do `peer1`.

Uma máquina de cada site virtual é utilizada para iniciar a aplicação. Essas máquinas executam o componente `MyGrid` que acessa o `peer` do mesmo site virtual ao qual elas pertencem. O restante das máquinas de cada um dos sites virtuais são identificadas como Máquinas da Grade (`GuMs` na terminologia do OurGrid) e executam o `UserAgent`, componente do OurGrid que habilita a execução de tarefas da grade. Nenhum outro componente do OurGrid foi utilizado na execução dos testes neste trabalho.

Três usuários, cada um localizado em um site virtual, executam uma aplicação. Esta é composta de 20 tarefas, divididas em três etapas: cópia de 1MB de dados para o nó, execução de uma pausa de 10 minutos e cópia de 1MB de dados de volta à máquina do usuário que iniciou a aplicação. No intuito de aumentar o tráfego de rede, a execução da aplicação foi configurada para que as tarefas fossem executadas somente em sites virtuais remotos. Por exemplo, tarefas da aplicação inicializada no site virtual 1 foram executadas somente nos sites 2 e 3. Todas as aplicações foram inicializadas ao mesmo tempo, cada uma a partir de um dos sites.

No intuito de demonstrar a efetividade do controle de rede, executaram-se testes de comunicação no ambiente de grade proposto nesta avaliação. Conforme apresentado anteriormente, a aplicação executada no ambiente copia dados entre a máquina do usuário e os nós da grade que executam as tarefas. Levando em consideração que as tarefas foram executadas somente nos sites virtuais remotos, criaram-se dois casos de controle de largura de banda entre os `peers`, sobre os quais os experimentos foram realizados. Um dos casos utiliza a largura total da banda disponível, no caso $100MBit/s$. No segundo caso é feito o controle de largura de banda em $1MBit/s$. O controle foi feito utilizando as ferramentas apresentadas na seção 4.3 através da arquitetura de gerência de rede de máquinas virtuais. A Tabela 6 apresenta a média de 20 (vinte) execuções em milissegundos para envio e recebimento dos dados de cada tarefa para os dois cenários de configuração de banda.

Tabela 6 – Tempo em milisegundos por tarefa para transmissão do arquivo.

Bandwidth configurada	Envio	Desvio Padrão	Recebimento	Desvio Padrão
1Mbit/s	136346	3381	113167	2806
100Mbit/s	5197	129	3463	86

Analisando a tabela é possível perceber e eficiência do controle de rede através da comparação dos tempos de comunicação entre dois cenários apresentados. Embora a diferença do valor de vazão configurado entre os dois cenários seja de 100 vezes, os valores de comunicação obtidos não acompanharam a mesma proporcionalidade. Ou seja, a transferência de 1MB de dados da tarefa no cenário de $100MBit/s$ deveria ocorrer em $1000ms$, já no cenário de $1MBit/s$ deveria ocorrer em $100000ms$.

Esse comportamento não foi observado devido aos efeitos de colisão em um ambiente de rede *Fast Ethernet* causados pela execução simultânea de diversas tarefas da aplicação, o que também é comum em ambientes reais. Além do mais, em um ambiente virtualizado existe o *overhead* no compartilhamento de recursos físicos, neste caso a interface de rede. Ambos os fatores influenciam no desempenho da comunicação de rede.

5.4 Considerações deste Capítulo

Em ambientes distribuídos, principalmente os de larga escala intermediados pela Internet, existem variações nos elementos de comunicação como a largura de banda e a latência de rede. Os testes realizados e apresentados neste capítulo demonstram a viabilidade de utilizar a virtualização de hardware, combinado com ferramentas de controle de rede, para emular ambientes distribuídos facilmente escaláveis.

Com a utilização da arquitetura de gerência de rede de máquinas virtuais e do *framework* apresentado na seção 5.2, pode-se configurar um ambiente onde quatro máquinas reais hospedaram um ambiente de grade computacional de 32 nós (Máquinas Virtuais). Neste ambiente foi executada uma aplicação que se comportou da mesma forma como em um ambiente somente com máquinas reais. Neste ambiente também foi avaliada a efetividade, focada na emulação de ambiente distribuídos, do controle de largura de banda e latência utilizando ferramentas comuns de rede.

6 Conclusão

A criação de ambientes virtuais utilizando virtualização de hardware é uma alternativa onde com poucos recursos computacionais se pode criar ambientes controlados e escaláveis. Dependendo da utilização desses ambientes, em alguns casos é necessária a configuração dos recursos de rede de forma a garantir questões como isolamento, controle de vazão e latência de rede.

Neste trabalho, foi feita uma pesquisa na intenção de utilizar os recursos de virtualização, focado na gerência de componentes de rede, para prover ambientes onde tais recursos precisam ser controlados. Assim, para automatizar a gerência dos recursos de rede, desenvolveu-se uma arquitetura de gerência onde questões como controle de vazão, latência e isolamento de rede foram estudados. A arquitetura recebe como entrada a descrição de dois ambientes: o real, que descreve a estrutura de máquinas e suas conexões de rede; e o virtual, que descreve a estrutura de rede que se deseja configurar entre as máquinas virtuais. Utilizando ferramentas como o TBF/NetEm e o *iptables*, combinadas com os recursos do sistema de virtualização, foi possível criar sub-redes de máquinas virtuais com controle de vazão e latência entre suas conexões.

Para validar a arquitetura de gerência de rede no sistema de virtualização, foi utilizado como contexto um *framework* de criação de ambientes para avaliação de aplicações distribuídas. Este *framework* criou um ambiente de máquinas virtuais e, com o auxilio da arquitetura proposta neste trabalho, configurou os aspectos de rede de um ambiente de *Grid*. Neste ambiente, organizou-se um conjunto de testes que demonstrou a possibilidade da execução de uma aplicação real em um ambiente utilizando tecnologias de virtualização. Então, pode-se concluir que é possível criar um ambiente distribuído utilizando máquinas virtuais e que recursos de rede são configurados de forma a atender o comportamento especificado como entrada da arquitetura.

A arquitetura de gerência de rede apresentada neste trabalho, embora desenvolvida para configurar diferentes sistemas de virtualização, foi validada sobre o Monitor de

Máquinas Virtuais Xen.

Em trabalhos futuros também esperamos fazer testes de configuração de componentes de rede em outros sistemas de virtualização, como o VMWare [29] e KVM [27]. Além disso, também pode ser avaliada a possibilidade de a arquitetura fazer configurações em tempo de execução, visto que em ambientes reais seus componentes são instáveis e podem sofrer alterações em vários momentos da execução das aplicações.

Referências

[1] CREASY, R. J. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, IBM, v. 25, n. 5, p. 483–490, Set 1981.

[2] MICROSYSTEMS, S. *Linguagem Java*. Disponível em: <http://java.sun.com>. Acesso em: 2 jul 2006.

[3] ROSENBLUM, M. The reincarnation of virtual machines. *ACM Queue*, ACM, v. 2, n. 5, p. 34–40, jul 2004.

[4] ROSENBLUM, M.; GARFINKEL, T. Virtual machine monitors: Current technology and future trends. *IEEE Computer*, IEEE Computer Society, v. 38, n. 5, p. 39–47, mai 2005.

[5] PLESSL, C.; PLATZNER, M. Virtualization of hardware – introduction and survey. In: *Proceedings of the 4th International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*. Las Vegas: CSREA, 2004. p. 63–69.

[6] MENASCÉ, D. A. Virtualization: Concepts, applications, and performance modeling. In: *Proceedings of 31th International Computer Measurement Group Conference*. Orlando: CMG, 2005. p. 407–414.

[7] MEYER, R. A.; SEAWRIGHT, L. H. A virtual machine time-sharing system. *IBM Systems Journal*, IBM, v. 9, n. 3, p. 199–218, jul 1970.

[8] SWSOFT. *Welcome to OpenVZ - Server Virtualization Open Source Project*. Disponível em: <http://openvz.org/>. Acesso em: 20 abr. 2007.

[9] VIRTUOZZO. *Virtualization, Virtual Server, Windows, Linux, Consolidation, Physical to Virtual, Disaster Recovery Software*. Disponível em: <http://www.virtuozzo.com/>. Acesso em: 27 set. 2006.

[10] SUSANTA, N.; TZI-CKER, C. *A Survey on Virtualization Technologies*. State University of New York, NY, 2007. Disponível em: <http://www.ecsl.cs.sunysb.edu/tr/TR179.pdf>. Acesso em: 28 abr. 2007.

[11] Microsoft Inc. *.NET Homepage*. Disponível em: <http://www.microsoft.com/net/>. Acesso em: 12 set. 2006.

[12] SYLOR, M.; TALLMAN, O. Applying network management standards to system management; the case for the common agent. In: *Proceedings of the IEEE First International Workshop on System Management*. Los Angeles: IEEE Computer Society, 1993. p. 110–117.

[13] STALLINGS, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. 3. ed. Reading: Addison Wesley, 1999.

[14] THOMPSON, J. Web-based enterprise management architecture. *IEEE Communications Magazine*, IEEE, v. 36, n. 3, p. 80–86, mar 1998.

[15] DMTF. *Specification for the Representation of CIM in XML Version 2.0*. Jul 1999. DMTF Especification. Disponível em: <http://www.dmtf.org/standards/wbem/CIM-XML>. Acesso em: 28 abr. 2007.

[16] DMTF. *Common Information Model (CIM)*. Disponível em: <http://www.dmtf.org/standards/standard_cim.php>. Acesso em: 21 abr. 2006.

[17] DMTF. *WBEM: Web Services for Management*. Disponível em: <http://www.dmtf.org/standards/wbem/wsman>. Acesso em: 21 abr. 2007.

[18] BARHAM, P. et al. Xen and the art of virtualization. In: *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. Bolton Landing, NY, USA: ACM, 2003. p. 164–177.

[19] LU, S. et al. On the consistency of XML DTDs. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 52, n. 2, p. 231–247, Mai 2005.

[20] RODRIGUES, G. *vMIB: uma MIB genérica para gerenciamento de recursos virtuais*. 2008. Dissertação (Mestrado em Ciência da Computação) — Pontifícia Universidade Católica do Rio Grande do Sul.

[21] CAMBRIDGE. *Página do Projeto Xen - Universidade de Cambridge*. Disponível em: <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>. Acesso em: 20 mar. 2006.

[22] WELTE, H. *What is NetFilter/IpTables?* Disponível em: <http://www.netfilter.org>. Acesso em: 28 set. 2006.

[23] POSTEL, J. *RFC 793: Transmission Control Protocol*. Set 1981. Disponível em: <ftp://ftp.math.utah.edu/pub/rfc/rfc793.txt>. Acesso em: 28 abr. 2007.

[24] ALMESBERGER, W. Linux traffic control — implementation overview. In: *Proceedings of 5th Annual Linux Expo*. Raleigh, North Carolina: Specialized Systems Consultants, Inc., 1999. p. 153–164.

[25] HEMMINGER, S. *NetEm*. Disponível em: <http://linux-net.osdl.org/index.php/Netem>. Acesso em: 20 abr. 2007.

[26] KUZNETSOV, A. N. *Token Bucket Filter - Linux Man Page*. Disponível em: <http://linux.die.net/man/8/tc-tbf>. Acesso em: 20 abr. 2007.

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

[28] DIKE, J. *User-mode Linux*. Indianapolis, Indiana: Prentice Hall, 2006.

[29] WALTERS, B. VMware virtual platform. *Linux Journal*, Specialized Systems Consultants, Inc., Seattle, WA, USA, v. 1999, n. 63es, p. 6, Ago 1999.

[30] ZIMMER, D. *VMware and Microsoft Virtual Server: virtuelle Server im professionellen Einsatz; [VMware GSX, ESX und Microsoft Virtual Server; Virtualisierungssoftware im Vergleich; Planung, Installation und Verwaltung]*. Bonn, Germany: Galileo Press, 2005. 612 p.

[31] VMWare Inc. *VMware VirtualCenter for VMware Server*. Disponível em: <http://www.vmware.com/files/pdf/virtualcenter_server_datasheet.pdf>. Acesso em: 15 set. 2007.

[32] YLONEN, T.; LONVICK, C. *The Secure Shell (SSH) Protocol Architecture*. jan 2006. IETF RFC 4251. Disponível em: <http://www.ietf.org/rfc/rfc4251.txt>. Acesso em: 28 abr. 2007.

[33] POSTEL, J. *Internet Control Message Protocol*. Set 1981. IETF RFC 792. Disponível em: <http://www.rfc-editor.org/rfc/rfc792.txt>. Acesso em: 28 abr. 2007.

[34] FRANCIOSI, F. et al. Deploying and Managing Xen Sites with XSM. In: *Proceedings of Workshop on Virtualization/Xen in HPC Cluster and Grid Computing Environments*. Rennes, França: Springer, 2007. v. 4854, p. 49–58.

[35] CIRNE, W. et al. Labs of the world, unite!!! *Journal of Grid Computing*, Springer, v. 4, n. 3, p. 225–246, Set 2006.

Apêndice A – Descrição de tipo dos arquivos *XML* da arquitetura

Arquivo de descrição de tipo de documento (DTD) ambiente real

```
<!ELEMENT RealNetwork (SubNet+)>

<!ELEMENT SubNet (NetworkLink+,RealMachine+)>

<!ELEMENT NetworkLink EMPTY>
<!ATTLIST NetworkLink router CDATA "">

<!ELEMENT RealMachine (NetworkInterface+, VirtualMachine+)>
<!ATTLIST RealMachine name CDATA #REQUIRED>

<!ELEMENT NetworkInterface (Ip,NetMask,Broadcast,Gateway,
                            SNMPif,Bandwidth,Latency)>
<!ATTLIST NetworkInterface id CDATA #IMPLIED>
<!ATTLIST NetworkInterface mac CDATA #REQUIRED>
<!ATTLIST NetworkInterface vif CDATA #IMPLIED>

<!ELEMENT Ip (#PCDATA)>

<!ELEMENT NetMask (#PCDATA)>

<!ELEMENT Broadcast (#PCDATA)>

<!ELEMENT Gateway (#PCDATA)>

<!ELEMENT SNMPif (#PCDATA)>

<!ELEMENT Bandwidth (#PCDATA)>
```

```
<!ELEMENT Latency (#PCDATA)>

<!ELEMENT VirtualMachine (VirtualNetworkInterface*)>
<!ATTLIST VirtualMachine name CDATA #REQUIRED>

<!ELEMENT VirtualNetworkInterface EMPTY>
<!ATTLIST VirtualNetworkInterface mac CDATA #REQUIRED>
<!ATTLIST VirtualNetworkInterface vif CDATA #REQUIRED>
```

Exemplo de descrição de um ambiente real

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RealNetwork SYSTEM "realEnvironment.dtd">

<RealNetwork>
  <SubNet>
    <NetworkLink router="0" />
    <RealMachine name="xen02">
      <NetworkInterface id="eth0" mac="00:12:79:64:8A:46">
        <Ip>192.168.5.102</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Broadcast>192.168.5.255</Broadcast>
        <Gateway>192.168.5.254</Gateway>
        <SNMPif>true</SNMPif>
        <Bandwidth>1.0E8</Bandwidth>
        <Latency>0.0010</Latency>
      </NetworkInterface>
      <VirtualMachine name="vmgrid107">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:07" vif="vif8.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid106">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:06" vif="vif7.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid105">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:05" vif="vif6.0" />
```

```
  </VirtualMachine>
  <VirtualMachine name="vmgrid104">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:04" vif="vif5.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid103">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:03" vif="vif4.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgridpeer1">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:01" vif="vif2.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid102">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:02" vif="vif3.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid101">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:00" vif="vif1.0" />
  </VirtualMachine>
</RealMachine>
<RealMachine name="xen01">
  <NetworkInterface id="eth0" mac="00:12:79:64:89:09">
    <Ip>192.168.5.101</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Broadcast>192.168.5.255</Broadcast>
    <Gateway>192.168.5.254</Gateway>
    <SNMPif>true</SNMPif>
    <Bandwidth>1.0E8</Bandwidth>
    <Latency>0.0010</Latency>
  </NetworkInterface>
  <VirtualMachine name="vmgrid204">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:0e" vif="vif15.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid108">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:08" vif="vif9.0" />
```

```
    </VirtualMachine>
    <VirtualMachine name="vmgrid203">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:0d" vif="vif14.0" />
    </VirtualMachine>
    <VirtualMachine name="vmgrid202">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:0c" vif="vif13.0" />
    </VirtualMachine>
    <VirtualMachine name="vmgrid201">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:0a" vif="vif11.0" />
    </VirtualMachine>
    <VirtualMachine name="vmgridpeer2">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:0b" vif="vif12.0" />
    </VirtualMachine>
    <VirtualMachine name="vmgrid205">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:0f" vif="vif16.0" />
    </VirtualMachine>
    <VirtualMachine name="vmgrid109">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:09" vif="vif10.0" />
    </VirtualMachine>
  </RealMachine>
  <RealMachine name="xen08">
    <NetworkInterface id="eth0" mac="00:12:79:64:8A:42">
      <Ip>192.168.5.108</Ip>
      <NetMask>255.255.255.0</NetMask>
      <Broadcast>192.168.5.255</Broadcast>
      <Gateway>192.168.5.254</Gateway>
      <SNMPif>true</SNMPif>
      <Bandwidth>1.0E8</Bandwidth>
      <Latency>0.0010</Latency>
    </NetworkInterface>
    <VirtualMachine name="vmgrid308">
      <VirtualNetworkInterface
       mac="00:12:13:14:00:1f" vif="vif32.0" />
```

```
  </VirtualMachine>
  <VirtualMachine name="vmgrid212">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:16" vif="vif23.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid211">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:15" vif="vif22.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid210">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:14" vif="vif21.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid209">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:13" vif="vif20.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid208">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:12" vif="vif19.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid207">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:11" vif="vif18.0" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid206">
    <VirtualNetworkInterface
     mac="00:12:13:14:00:10" vif="vif17.0" />
  </VirtualMachine>
</RealMachine>
<RealMachine name="xen07">
  <NetworkInterface id="eth0" mac="00:12:79:64:89:7F">
    <Ip>192.168.5.107</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Broadcast>192.168.5.255</Broadcast>
    <Gateway>192.168.5.254</Gateway>
    <SNMPif>true</SNMPif>
    <Bandwidth>1.0E8</Bandwidth>
    <Latency>0.0010</Latency>
```

```
      </NetworkInterface>
      <VirtualMachine name="vmgridpeer3">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:18" vif="vif25.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid307">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:1e" vif="vif31.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid306">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:1d" vif="vif30.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid305">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:1c" vif="vif29.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid304">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:1b" vif="vif28.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid303">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:1a" vif="vif27.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid302">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:19" vif="vif26.0" />
      </VirtualMachine>
      <VirtualMachine name="vmgrid301">
        <VirtualNetworkInterface
         mac="00:12:13:14:00:17" vif="vif24.0" />
      </VirtualMachine>
    </RealMachine>
  </SubNet>
</RealNetwork>
```

Arquivo de descrição de tipo de documento (DTD) para descrição do ambiente virtual

```
<!ELEMENT VirtualNetwork (SubNet+)>

<!ELEMENT SubNet (NetworkLink+,VirtualMachine+)>
<!ATTLIST SubNet net CDATA #REQUIRED>
<!ATTLIST SubNet netmask CDATA "">


<!ELEMENT NetworkLink EMPTY>
<!ATTLIST NetworkLink router CDATA "">
<!ATTLIST NetworkLink bandwidth CDATA "">
<!ATTLIST NetworkLink latency CDATA "">

<!ELEMENT VirtualMachine (NetworkInterface*,Route*)>
<!ATTLIST VirtualMachine name CDATA #REQUIRED>

<!ELEMENT NetworkInterface (Ip,NetMask,Gateway,Broadcast)>
<!ATTLIST NetworkInterface id CDATA #IMPLIED>
<!ATTLIST NetworkInterface mac CDATA #REQUIRED>

<!ELEMENT Ip (#PCDATA)>

<!ELEMENT NetMask (#PCDATA)>

<!ELEMENT Broadcast (#PCDATA)>

<!ELEMENT Gateway (#PCDATA)>

<!ELEMENT Route EMPTY>
<!ATTLIST Route net CDATA #REQUIRED>
<!ATTLIST Route gateway CDATA #REQUIRED>
```

Exemplo de descrição de um ambiente virtual

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE VirtualNetwork SYSTEM "virtualEnvironment.dtd">

<VirtualNetwork>
  <SubNet net="192.168.7.0" netmask="255.255.255.0">
    <NetworkLink bandwidth="10000.0" latency="0.01" router="0" />
    <VirtualMachine name="vmgrid308">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:1f">
        <Ip>192.168.7.9</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.7.254</Gateway>
        <Broadcast>192.168.7.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid307">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:1e">
        <Ip>192.168.7.8</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.7.254</Gateway>
        <Broadcast>192.168.7.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid306">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:1d">
        <Ip>192.168.7.7</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.7.254</Gateway>
        <Broadcast>192.168.7.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid305">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:1c">
        <Ip>192.168.7.6</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.7.254</Gateway>
        <Broadcast>192.168.7.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid304">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:1b">
```

```
    <Ip>192.168.7.5</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.7.254</Gateway>
    <Broadcast>192.168.7.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid303">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:1a">
    <Ip>192.168.7.4</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.7.254</Gateway>
    <Broadcast>192.168.7.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid302">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:19">
    <Ip>192.168.7.3</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.7.254</Gateway>
    <Broadcast>192.168.7.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid301">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:17">
    <Ip>192.168.7.2</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.7.254</Gateway>
    <Broadcast>192.168.7.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgridpeer3">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:18">
    <Ip>192.168.7.1</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.7.254</Gateway>
    <Broadcast>192.168.7.255</Broadcast>
  </NetworkInterface>
  <Route net="192.168.6.0" gateway="192.168.7.1" />
  <Route net="192.168.5.0" gateway="192.168.7.1" />
```

```
    </VirtualMachine>
  </SubNet>
  <SubNet net="192.168.6.0" netmask="255.255.255.0">
    <NetworkLink bandwidth="10000.0" latency="0.01" router="0" />
    <VirtualMachine name="vmgrid210">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:14">
        <Ip>192.168.6.11</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgridpeer2">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:0b">
        <Ip>192.168.6.1</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
      <Route net="192.168.7.0" gateway="192.168.6.1" />
      <Route net="192.168.5.0" gateway="192.168.6.1" />
    </VirtualMachine>
    <VirtualMachine name="vmgrid209">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:13">
        <Ip>192.168.6.10</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid208">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:12">
        <Ip>192.168.6.9</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid207">
```

```
  <NetworkInterface id="eth0" mac="00:12:13:14:00:11">
    <Ip>192.168.6.8</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.6.254</Gateway>
    <Broadcast>192.168.6.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid206">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:10">
    <Ip>192.168.6.7</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.6.254</Gateway>
    <Broadcast>192.168.6.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid205">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:0f">
    <Ip>192.168.6.6</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.6.254</Gateway>
    <Broadcast>192.168.6.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid204">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:0e">
    <Ip>192.168.6.5</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.6.254</Gateway>
    <Broadcast>192.168.6.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid203">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:0d">
    <Ip>192.168.6.4</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.6.254</Gateway>
    <Broadcast>192.168.6.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
```

```
    <VirtualMachine name="vmgrid202">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:0c">
        <Ip>192.168.6.3</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid201">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:0a">
        <Ip>192.168.6.2</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid212">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:16">
        <Ip>192.168.6.13</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
    <VirtualMachine name="vmgrid211">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:15">
        <Ip>192.168.6.12</Ip>
        <NetMask>255.255.255.0</NetMask>
        <Gateway>192.168.6.254</Gateway>
        <Broadcast>192.168.6.255</Broadcast>
      </NetworkInterface>
    </VirtualMachine>
  </SubNet>
  <SubNet net="192.168.5.0" netmask="255.255.255.0">
    <NetworkLink bandwidth="10000.0" latency="0.01" router="0" />
    <VirtualMachine name="vmgrid105">
      <NetworkInterface id="eth0" mac="00:12:13:14:00:05">
        <Ip>192.168.5.6</Ip>
        <NetMask>255.255.255.0</NetMask>
```

```
    <Gateway>192.168.5.254</Gateway>
    <Broadcast>192.168.5.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid104">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:04">
    <Ip>192.168.5.5</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.5.254</Gateway>
    <Broadcast>192.168.5.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid103">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:03">
    <Ip>192.168.5.4</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.5.254</Gateway>
    <Broadcast>192.168.5.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid102">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:02">
    <Ip>192.168.5.3</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.5.254</Gateway>
    <Broadcast>192.168.5.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgrid101">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:00">
    <Ip>192.168.5.2</Ip>
    <NetMask>255.255.255.0</NetMask>
    <Gateway>192.168.5.254</Gateway>
    <Broadcast>192.168.5.255</Broadcast>
  </NetworkInterface>
</VirtualMachine>
<VirtualMachine name="vmgridpeer1">
  <NetworkInterface id="eth0" mac="00:12:13:14:00:01">
    <Ip>192.168.5.1</Ip>
```

```
      <NetMask>255.255.255.0</NetMask>
      <Gateway>192.168.5.254</Gateway>
      <Broadcast>192.168.5.255</Broadcast>
    </NetworkInterface>
    <Route net="192.168.6.0" gateway="192.168.5.1" />
    <Route net="192.168.7.0" gateway="192.168.5.1" />
  </VirtualMachine>
  <VirtualMachine name="vmgrid109">
    <NetworkInterface id="eth0" mac="00:12:13:14:00:09">
      <Ip>192.168.5.10</Ip>
      <NetMask>255.255.255.0</NetMask>
      <Gateway>192.168.5.254</Gateway>
      <Broadcast>192.168.5.255</Broadcast>
    </NetworkInterface>
  </VirtualMachine>
  <VirtualMachine name="vmgrid108">
    <NetworkInterface id="eth0" mac="00:12:13:14:00:08">
      <Ip>192.168.5.9</Ip>
      <NetMask>255.255.255.0</NetMask>
      <Gateway>192.168.5.254</Gateway>
      <Broadcast>192.168.5.255</Broadcast>
    </NetworkInterface>
  </VirtualMachine>
  <VirtualMachine name="vmgrid107">
    <NetworkInterface id="eth0" mac="00:12:13:14:00:07">
      <Ip>192.168.5.8</Ip>
      <NetMask>255.255.255.0</NetMask>
      <Gateway>192.168.5.254</Gateway>
      <Broadcast>192.168.5.255</Broadcast>
    </NetworkInterface>
  </VirtualMachine>
  <VirtualMachine name="vmgrid106">
    <NetworkInterface id="eth0" mac="00:12:13:14:00:06">
      <Ip>192.168.5.7</Ip>
      <NetMask>255.255.255.0</NetMask>
      <Gateway>192.168.5.254</Gateway>
      <Broadcast>192.168.5.255</Broadcast>
    </NetworkInterface>
  </VirtualMachine>
```

```
  </SubNet>
</VirtualNetwork>
```