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

vMIB: Uma MIB Genérica para Gerenciamento de Recursos Virtuais

Guilherme da Cunha Rodrigues

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

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

Porto Alegre
2008

Pontifícia Universidade Católica do Rio Grande do Sul

Dados Internacionais de Catalogação na Publicação (CIP)

R696v Rodrigues, Guilherme da Cunha
vMIB : uma MIB genérica para gerenciamento de recursos virtuais / Guilherme da Cunha Rodrigues. – Porto Alegre, 2008.
104 f.

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

1. Informática. 2. Redes de Computadores – Gerência. 3. Protocolos de Gerenciamento. 4. Comunicação de Dados. I. Título.

CDD 004.62

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

PUCRS
Campus Central
Av. Ipiranga, 6681 - prédio 16 - CEP 90619-900
Porto Alegre - RS - Brasil
Fone: +55 (51) 3320-3544 - Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

Para minha família.

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 "**vMIB: Uma MIB Genérica para Gerenciamento de Recursos Virtuais**", apresentada por Guilherme da Cunha Rodrigues, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Ciência da Computação, aprovada em 17/01/08 pela Comissão Examinadora:

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

Prof. Dr. Fernando Luís Dotti – — PPGCC/PUCRS

Prof. Dr. Luciano Paschoal Gaspary – — UFRGS

Homologada em...../...../....., conforme Ata No. ..... 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

Agradecimentos

A Deus.
A Minha família.
Ao Professor César Augusto F. De Rose.
Aos Colegas Rodrigo Calheiros, Mauro Storch, Rafael Antonioli e Fábio Rossi.
A todas as pessoas que participaram do meu processo de crescimento humano e profissional.
Meu Muito Obrigado a Todos !!!
E saibam que podem sempre contar comigo também.
Por enquanto .....era isso.....abraço!!!

Resumo

A tecnologia de virtualização permite que várias máquinas virtuais possam ser criadas sobre uma mesma estrutura de hardware. Com o crescimento da utilização desta tecnologia, a demanda por métodos mais eficientes de administração desse tipo de recurso também cresce. Atualmente as ferramentas disponíveis para gerenciamento de sistemas não possuem suporte adequado para gerenciar sistemas virtuais eficientemente, pois não consideram fatores pertinentes aos recursos virtualizados, tais como processador, memória, disco e rede.

O desenvolvimento de uma MIB genérica proporciona uma ferramenta eficaz para o gerenciamento de máquinas virtuais, que agrega dentre outras vantagens uma maior flexibilidade à atividade de gerência , bem como a capacidade de interoperar o gerenciamento de diferentes sistemas virtuais.

Este trabalho apresenta uma MIB genérica utilizada para o gerenciamento de máquinas virtuais e monitores de máquinas virtuais. Esta MIB virtual (vMIB) foi criada com base nos padrões definidos pela SMI (*Structure Management Information*) e no modelo de referência da MIB-II, a vMIB foi validada utilizando o protocolo SNMP (*Simple Network Management Protocol*).
Palavras-chave: Virtualização, Gerência, Máquinas Virtuais.

Abstract

Virtualization technology allows several virtual machines to be created from a single hardware structure. With virtualized computational systems increasing utilization, demand for efficient management of this sort of resource also grows. Network management tools currently available do not manage virtual systems efficiently, because specific issues related to virtual resources, such as processor, memory, disk and network, are not considered.

Development of a generic MIB not only provides an efficient tool for virtual machines management but also adds other advantages, such as more management flexibility as well as interoperability among different virtual systems.

This work proposes a independent MIB to be used for management of both virtual machines and virtual machine monitors. This virtual MIB (vMIB) was created based on the structures of MIB-II and SMI (*Structure Management Information*) specification and was validated using the SNMP (*Simple Network Management Protocol*) protocol.
Keywords: Virtualization, Management, Virtual Machines.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AT	*Address Translation*	37
BIOS	*Basic Input/Output System*	27
CPU	*Central Processing Unit*	24
DTMF	*Distributed Management Task Force*	24
EGP	*External Gateway Protocol*	38
ICMP	*Internet Control Message Protocol*	38
ID	*Identification*	68
IP	*Internet Protocol*	24
MAC	*Media Access Control*	41
MIB	*Management Information Base*	24
NA	*Not Accessible*	37
OSI	*Open Systems Interconnection*	33
RO	*Read Only*	37
RW	*Read Write*	37
SMI	*Structure Management Information*	35
SNMP	*Simple Network Management Protocol*	24
TCP	*Transfer Control Protocol*	24
UDP	*User Datagram Protocol*	24
VBD	*Virtual Block Devices*	29
VE	*Virtual Environment*	31
VIF	*Virtual Network Interface*	29
VM	*Virtual Machine*	22
vMIB	*virtual MIB*	61
VMM	*Virtual Machine Monitor*	22
VPS	*Virtual Private Server*	31
VRF	*Virtual Firewall Router*	29
WBEM	*Web-Based Enterprise Management*	24
XM	*Xen Management*	75

Sumário

1 Introdução

O conceito de virtualização não é novo e tem princípios na década de 60 com o VM/370 da IBM que já utilizava tal técnica [1]. Atualmente ela é empregada em diversas áreas da computação, as quais vão desde serviços de suporte que utilizam vários sistemas até a área de alto desempenho. Isto ocorre porque a virtualização melhora a escalabilidade do sistema de maneira geral [2]. Essa tendência tem fomentado o interesse de grandes empresas no desenvolvimento de tecnologias específicas para implementação de sistemas virtuais, dentre as quais podemos citar os esforços da Intel para desenvolver e lançar no mercado uma família de processadores adequados a virtualização [3].

A virtualização permite que um único computador (*host*) hospede um ou mais ambientes, proporcionando vantagens como melhor utilização dos recursos, consolidação dos servidores, segurança (através do isolamento entre diferentes VM's), tolerância a falhas e balanceamento de carga [4]. Uma das técnicas utilizadas para implementar virtualização é através da *paravirtualização*, onde a virtualização da plataforma de hardware é implementada através da inserção de um módulo de programa ao Kernel do sistema operacional que roda sobre o virtualizador [5].

Com o acelerado desenvolvimento de sistemas computacionais, a demanda por melhores métodos para administrá-los também aumenta significamente. A utilização e o amadurecimento da tecnologia de virtualização conseqüentemente também são afetados por tal tendência, visto que as peculiaridades pertinentes ao sistema virtualizado também devem ser consideradas quando gerenciamos tal tipo de ambiente. Para tanto, a utilização de protocolos e ferramentas de gerência de recursos torna-se um dispositivo importante para um melhor aproveitamento das máquinas virtuais.

Quando trabalhamos com máquinas virtuais, devemos considerar fatores pertinentes a cada máquina virtual e ao sistema real no qual a máquina está alocada como, por exemplo, o total de memória disponível e a quantidade de memória utilizada por cada máquina virtual. Tais fatores devem ser considerados também no momento em que é realizado controle e gerência do sistema, para tanto é necessária a utilização de recursos que tenham a capacidade de reconhecer dentro do escopo da monitoração o que se refere a recurso virtual e o que se refere a recurso real do sistema.

Uma base de informações serve como referência na definição dos recursos que são gerenciados, ou seja, uma base de informações utiliza-se desses objetos como referência aos componentes do sistema gerenciado, facilitando o acesso a informações e provendo desempenho e flexibilidade à tarefa de administrar sistemas computacionais, para tanto sua arquitetura de maneira geral deve ser bem elaborada de forma que os objetos que a compõem possam estar

organizados de modo padronizado e simples [6].

Atualmente a utilização de sistemas virtuais está novamente em crescimento devido à ampla quantidade de aplicações em tecnologia. Porém, de modo a otimizar a utilização da tecnologia e prover um gerenciamento dinâmico e eficaz desses recursos necessita-se de ferramentas as quais realizem o gerenciamento dos recursos que compõem os sistemas virtuais, para tanto se faz necessária a criação de dispositivos que realizem tal tarefa. A utilização de protocolos de gerência de recursos e suas bases de informações definem atualmente padrões de gerenciamento dos mais variados tipos de recursos computacionais, porém ainda não encontramos opções de uma solução portável e dinâmica para gerenciamento dos diversos tipos de sistemas que implementam virtualização.

O intuito desse trabalho é a elaboração de um modelo de base de informações que visa a trabalhar com máquinas virtuais, para tanto o estudo de tais bases de informações agregado ao conhecimento prévio sobre virtualização e máquinas virtuais torna-se necessário.

1.1 Máquinas Virtuais

Devemos entender sistemas computacionais como conjuntos de níveis de abstração, nos quais notadamente o hardware é o nível mais baixo e o núcleo do sistema é a camada responsável pelo gerenciamento do sistema [7]. Uma máquina virtual simula uma réplica do hardware da máquina real e os processos têm a ilusão de que possuem dispositivos físicos disponíveis somente para ele. Analisando de outra maneira, seria como se fosse instalado um sistema operacional e sobre ele se teria suporte para operar qualquer outro sistema operacional, podendo ser desde qualquer distribuição Linux até a família Windows.

O conceito de máquinas virtuais não é novo. Ele foi originalmente desenvolvido para ser utilizado no VM/370 da IBM no final da década de 60. Pode-se definir uma máquina virtual (VM) como uma máquina abstrata, que permite que a máquina real seja particionada de tal modo que diversos sistemas operacionais sejam executados ao mesmo tempo [8].

A máquina virtual funciona através de um monitor que executa a função de gerenciador dos ambientes virtuais que estão em execução. Esse monitor de máquinas virtuais cria várias máquinas virtuais que trabalham sem que nenhuma interfira na outra e também não tenham influência no sistema base no qual à máquina esta alocada. Portanto, cada máquina virtual opera com um sistema operacional independente, com acesso a dispositivos e ferramentas inerentes a um sistema normal.

1.2 Sistemas Virtuais

Atualmente dentre os sistemas que trabalham com máquinas virtuais podemos destacar os sistemas Xen, VMware, OpenVZ e o *User-Mode Linux* (UML).

O Xen é um paravirtualizador livre desenvolvido originalmente para a arquitetura IA-32, que atualmente suporta também x86/64, Itanium e PowerPC. Por se tratar de um paravirtualizador, os sistemas operacionais executando nas VMs devem ser adaptados para serem compatíveis com o VMM. No entanto, recentes versões do Xen suportam, por exemplo, o Windows XP sem modificações se for utilizado, sob o virtualizador, um processador com suporte a virtualização, como por exemplo, os processadores "Pacifica" da AMD [9] ou os processadores Intel com tecnologia "VT" [3].

O Xen VMM coordena a execução de uma ou mais VMs. As máquinas virtuais no Xen (também conhecidas como domínios) podem ser de dois tipos. O primeiro contém máquinas virtuais que executam aplicações de usuários e são conhecidos como *DomU* – de *unprivileged*. Essas máquinas não têm acesso direto ao hardware real da máquina. O segundo (*Domain 0* ou simplesmente *Dom0*) é responsável pela gerência das demais VMs e pelo acesso ao hardware da máquina. Apenas um *Dom0* executa em um *host*. Também é possível, através do *Dom0*, monitorar o consumo de recursos de cada uma das VMs [10].

Diferentemente do Xen, o VMware é um software que cria máquinas virtuais as quais simulam um computador completo, onde é permitindo instalar qualquer sistema operacional. Utilizando o VMware é possível executar várias máquinas virtuais simultaneamente e executar lado a lado várias versões do Linux ou do Windows [11]. O VMware utiliza-se do conceito de emulação para colocar em funcionamento suas máquinas virtuais. O emulador é um software que transcreve instruções de um processador alvo para o processador no qual ele está rodando, ele também deve realizar a simulação dos circuitos integrados do hardware em um software [12]. O VMware organiza as máquinas virtuais hospedadas no formato de arquivos que ficam hospedados no sistema VMware. Cada máquina virtual poderá possuir configurações distintas para seu sistema instalado, como disponibilidade de memória, quantidade de processadores virtuais, entre outras.

Uma terceira alternativa para implementar a virtualização é o sistema OpenVZ, semelhantemente ao Xen o OpenVZ implementa máquinas virtuais utilizando o conceito de paravirtualização utilizando Linux, o OpenVZ trabalha criando ambientes virtuais isolados que funcionam como máquinas convencionais, porém sobre a mesma estrutura de hardware [13].

1.3 Protocolos de Gerenciamento

Um protocolo de gerência de recursos tem o objetivo de monitorar e controlar sistemas computacionais.

Com o acelerado desenvolvimento de sistemas computacionais, a demanda por melhores métodos para administrá-los também aumenta significamente. A utilização e o amadurecimento da tecnologia de virtualização conseqüentemente também são diretamente afetados por tal tendência, visto que as peculiaridades pertinentes ao sistema virtualizado também devem ser consideradas quando gerenciamos tal ambiente.

Visando a um melhor aproveitamento dos recursos reais, a utilização de protocolos de gerência torna-se uma ferramenta importante. Atualmente dentre os protocolos de gerência existentes podemos destacar dois protocolos, o *Simple Network Management Protocol* (SNMP) e o *Web-Based Enterprise Manageament* (WBEM). Porém, cabe ressaltar que mesmo tratando-se de protocolos largamente utilizados os mesmos não possuem bases de informações adequadas para trabalhar com máquinas virtuais independentemente do sistema que implementa a virtualização.

O SNMP (*Simple Network Management Protocol*) é um protocolo originariamente desenvolvido para gerencia de redes de computadores. Porém, por sua flexibilidade pode ser utilizado para outros tipos de aplicações de gerência. A estrutura do protocolo é baseada em agentes e gerentes, os agentes estão espalhados em uma rede baseada na pilha de protocolos TCP/IP e os dados são obtidos através de requisições de um gerente a um ou mais agentes utilizando os serviços do protocolo UDP (*User Datagram Protocol*) para enviar e receber suas mensagens através da rede. As variáveis que podem ser requisitadas fazem parte de uma MIB (*Manageament Information Base*) [14]. A MIB consiste no conjunto dos objetos gerenciados, que visa a armazenar todas as informações pertinentes ao sistema gerenciado.

Diferentemente do SNMP, o WBEM (*Web-Based Enterprise Management*) é na realidade mais do que um simples protocolo. Ele é uma arquitetura de gerenciamento elaborada pela DTMF (*Distributed Management Task Force*). Visando a representar a informação, o padrão WBEM utiliza o CIM (*Common Information Model*). A tarefa de transporte no padrão WBEM é realizada através do *CIM-XML Encoding* com o intuito de codificar os dados CIM em dados XML. Posteriormente, os dados são de fato transportados utilizando o protocolo HTTP (*Hyper-Text Transfer Protocol*) [15], [16], [17], [18].

1.4 Base de Informações

Uma base de informações é um modelo abstrato de referência para o gerenciamento de objetos reais tais como memória, CPU, dispositivos de rede, etc.

Para melhor explicar seu funcionamento utilizaremos como exemplo a arquitetura da MIB-II, onde os objetos estão disponibilizados na base conforme a Figura 1. Nota-se que o caminho para construção de cada objeto segue um padrão hierárquico e um objeto é referenciado através de uma seqüência de números separados por pontos que indicam o caminho seguido para construção do mesmo. Tal caminho servirá como referência indicando a localização física do recurso gerenciado.

Como por exemplo, podemos referenciar o objeto 1.3.6.1.2.1.11, que conforme o padrão de nomenclatura da MIB-II mostra abstratamente a descrição do objeto que referencia o protocolo SNMP no sistema gerenciado.

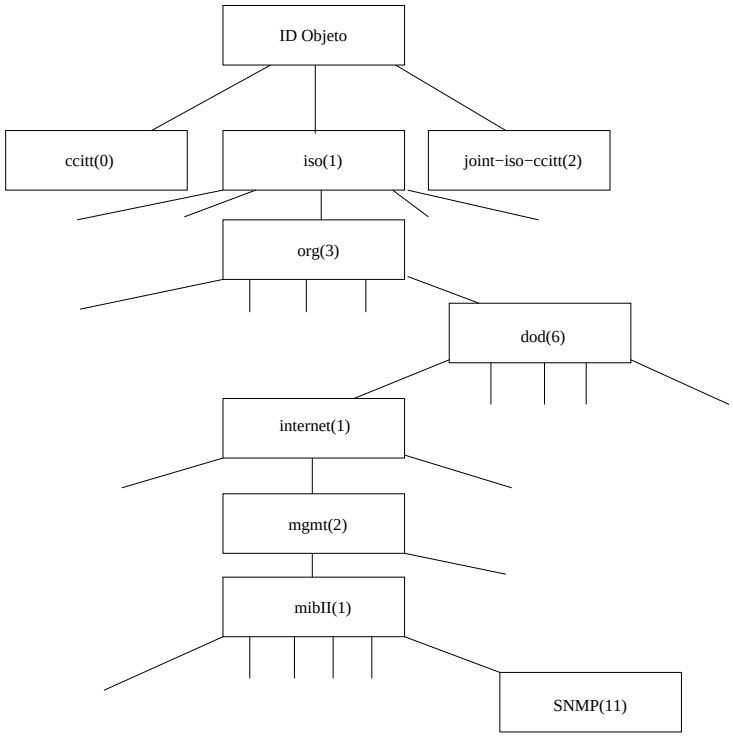

Figura 1 – Arquitetura da Mib II.

Pode-se destacar também que tal tipo de metodologia para construção de objetos provê recursos para se gerenciar ambientes heterogêneos, visto que o modelo hierárquico é flexível podendo abranger diversos sistemas, dispositivos, protocolos, etc, o que acrescenta interoperabilidade entre os sistemas gerenciados.Tornando-se assim uma ferramenta eficiente, independentemente de plataforma [19].

A utilização deste tipo de tecnologia de gerenciamento é largamente difundida e utilizada para gerência de redes, porém por possuir uma arquitetura flexível pode ser empregada para o gerenciamento de qualquer recurso computacional.

2 Sistemas Virtuais

Sistemas Virtuais são sistemas que possibilitam a utilização de dois ou mais sistemas operacionais sobre uma mesma estrutura de hardware. Atualmente destacam-se em dois grupos dispostos de acordo com a forma de implementação, virtualização e paravirtualização [20].

A virtualização é o processo de executar vários sistemas operacionais em um único equipamento. Uma máquina virtual é um ambiente operacional completo que se comporta como se fosse um computador independente. Com a virtualização, por exemplo, um servidor pode manter vários sistemas operacionais em uso em máquinas virtuais distintas. Isso acarreta em custo derivado do alto consumo de processamento, devido ao fato de possuir uma estrutura de hardware única utilizada pelo virtualizador. Cada requisição de hardware feita pelo sistema hospedeiro é entregue ao virtualizador, que se encarrega de fazer a transação com o hardware real. Entre os mais conhecidos encontram-se o Qemu e VMware. Nesses softwares emuladores de hardware, o sistema operacional é instalado em um arquivo, ou inicializado a partir de um arquivo [21].

Outra técnica utilizada para implementar a virtualização é a paravirtualização, na qual o software trabalha mais próximo do hardware real, pois um sistema operacional usando um kernel alterado gerencia sistemas paravirtualizados, operando sobre kernels alocado sobre as máquinas virtuais criadas. Com isso, ganha-se em desempenho, pois não há uma emulação completa desde a BIOS (*Basic Input/Output System*), possibilitando a execução de muitos sistemas operacionais simultaneamente [22].

Atualmente utilizando-se a paravirtualização destacam-se os sistemas Xen e OpenVZ. Nas seções seguintes, abordaremos especificamente os fundamentos dos sistemas Xen, VMware e OpenVZ, devido estes serem importantes no desenvolvimento desse trabalho.

2.1 Xen

O Xen é um paravirtualizador, *Virtual Machine Monitor* (VMM) ou *Hypervisor* , desenvolvido pela Universidade de Cambridge inicialmente para a arquitetura x86 [23]. O Xen tem a capacidade de gerenciar várias *Virtual Machines* alocadas sobre uma mesma estrutura física. E possui como principais características:

- Migração de máquinas virtuais entre *hosts*;
- Atualmente possui suporte para x86/32 e x86/64;

- Bom suporte de hardware;
- Suporte para tecnologia VT-x (*Intel Virtualization Technology*).

O Xen gerencia e organiza as requisições feitas pelas máquinas virtuais, validando-as, limitando-se apenas a repassar as instruções, sem interpretá-las como faria um emulador, por exemplo [24].

Particularmente no Xen, o sistema que vai ser executado dentro da máquina virtual, necessita ser modificado, assim como o sistema que funcionará no hospedeiro, ou seja, torna-se necessária também uma versão específica do sistema operacional para que se possa alterá-lo, instalando um *patch* no kernel antes de executá-lo [25].

O Xen prevê também o cumprimento de algumas premissas que foram analisadas para o seu desenvolvimento, destas podemos destacar o suporte a aplicações binárias sem a necessidade de compilação, a utilização do conceito de paravirtualização com o objetivo de ganho de desempenho, o isolamento dos recursos e a tentativa de manter o desempenho das máquinas virtuais hospedadas semelhante ao das máquinas reais.

O gerenciamento do sistema é realizado pelo chamado *dom0* ou *hypervisor*, que funciona como um supervisor das outras máquinas virtuais(*domU*) em execução, tal supervisor trabalha também realizando a interface entre os dispositivos de hardware e os sistemas operacionais instalados no *host*. Em um *host* podemos ter apenas um dom0 [26].

A arquitetura básica do Xen pode ser melhor observada na Figura 2.

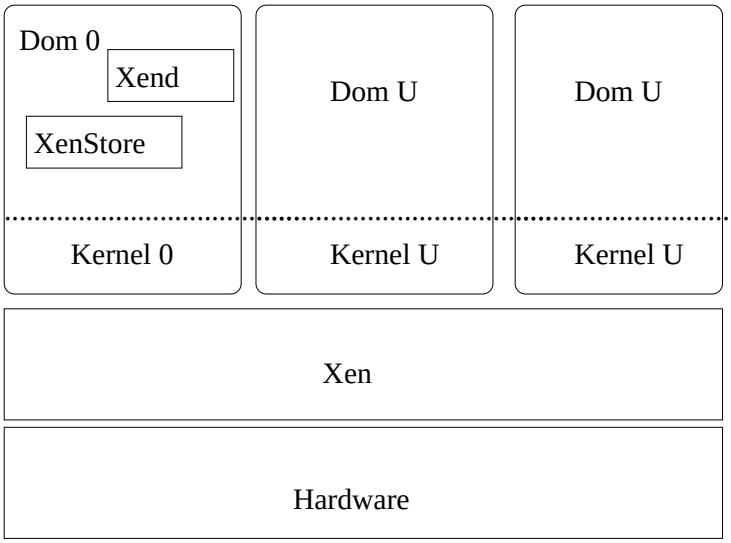

Figura 2 – Arquitetura Básica do Xen.

Analisando a Figura 2, nota-se na base da estrutura o nível de hardware interagindo com o Xen(Sistema Operacional alterado), dentro do *dom0* destaca-se o Xenstore, área destinada a armazenar os arquivos pertinentes ao Xen e o Xend, responsável pelo gerenciamento das máquinas virtuais, a direita observa-se as máquinas virtuais criadas.

Na concepção do Xen foi necessário também analisar alguns fatores complexos como gerenciamento de memória, de CPU, dispositivos de entrada e saída e da rede de comunicação.Os detalhes da interface são:

- Gerenciamento de Memória: a virtualização da memória representa uma grande dificuldade para o conceito de paravirtualização aplicado a uma arquitetura, pois os mecanismos requerem que o hypervisor efetue a alocação das áreas de memória a serem utilizadas por cada um dos sistemas hospedados, para tanto o sistema Xen realiza as atualizações das tabelas de memória, deixando o acesso somente para leitura, visando a melhoria de desempenho. O sistema operacional hospedado visualiza a sua porção de memória contínua mesmo que na pratica isto não aconteça. A cada domínio criado é alocado um tamanho de memória dentro de um limite máximo estipulado para cada domínio (conceito de *baloon*), se for necessário o sistema operacional hospedado pode requisitar um aumento de capacidade (aumento de *baloon*). Caso seja necessária a utilização de *swap*, cada sistema operacional hospedado realiza o seu gerenciamento sobre a proteção do *hypervisor*.

- Gerenciamento de CPU: a arquitetura x86 implementa camadas de privilégio genericamente descritas em camadas, enumeradas de 0 a 3, sendo que a camada 0 é a mais privilegiada e a 3 a menos. O Xen utilizando-se do hypervisor desloca a execução do sistema operacional hospedado para a camada 1, mantendo-se na camada 0 sob seu comando, com a finalidade de manter o controle das instruções privilegiadas, gerenciando assim a alocação, cabe ressaltar também que normalmente as aplicações de usuário rodam na camada 3, ou seja, a menos privilegiada.

- Dispositivos de Entrada e Saída: O Xen implementa um conjunto de abstrações simples para os dispositivos, permitindo projetar uma interface que seja eficiente e satisfatória quanto à proteção e ao isolamento. O acesso ao disco, por exemplo, é realizado através de uma destas abstrações chamadas de VBD (*Virtual Block Devices*) no qual existe uma para cada dispositivo físico, sendo controlada pelo hypervisor.

- Dispositivos de rede: Na abstração de rede existe o VFR (*Virtual Firewall Router*) que coordena as chamadas VIF(*Virtual Network Interface*), que seriam as placas de rede de cada sistema operacional hospedado.

Observando o Xen podemos concluir que é um sistema virtual que provê funcionalidade, performance e portabilidade no compartilhamento de sistemas operacionais comuns a um mesmo hardware.

2.2 VMware

O VMware é um software que cria máquinas virtuais que simulam um PC completo, no qual é permitido instalar praticamente qualquer sistema operacional [27]. Utilizando o VMware é possível executar várias máquinas virtuais simultaneamente e rodar lado a lado várias versões do Linux ou do Windows, por exemplo.

Contrariamente ao Xen, o VMware utiliza-se do conceito de emulação para colocar em funcionamento suas máquinas virtuais. O VMware organiza as máquinas virtuais hospedadas no formato de arquivos que ficam hospedados no sistema VMware. Cada máquina virtual poderá possuir configurações distintas para seu sistema instalado, como disponibilidade de memória, quantidade de processadores virtuais, entre outras. Sendo que essas configurações podem variar de máquina virtual para máquina virtual, ou seja, podemos ter um sistema Windows que possui 2 processadores virtuais e 512Mb e ao mesmo tempo uma distribuição Linux que possui 1 processador virtual e 256Mb, por exemplo.

Analisando o VMware observa-se que no nível de funcionamento ele trata cada máquina virtual como um arquivo, com isto, pode-se até mesmo levar cópias de máquinas virtuais de um micro para outro, um fator que merece destaque também é a possibilidade de comunicação entre as máquinas virtuais como se estivessem em uma rede tradicional [28].

Pode-se notar na Figura 3 o funcionamento do VMware, onde observamos a aplicação do conceito de emulação para criar sistemas operacionais virtuais.

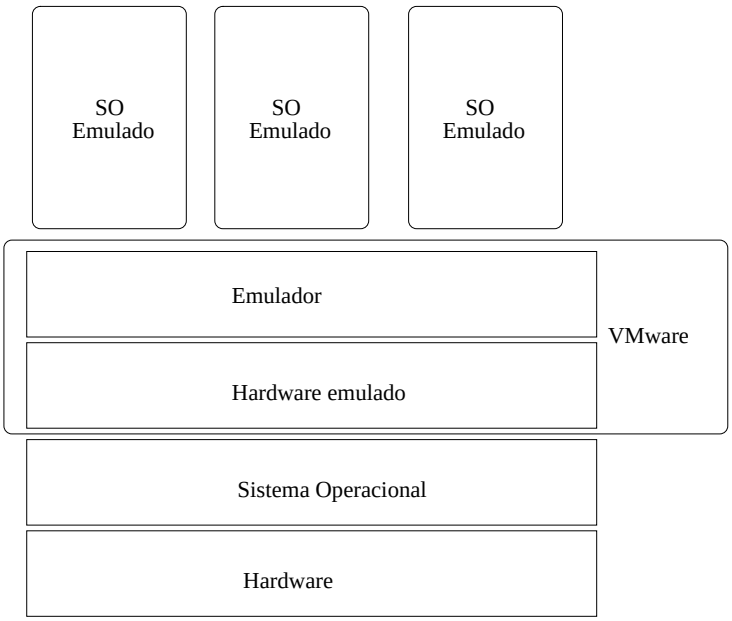

Figura 3 – Arquitetura Básica do VMware.

2.3 OpenVZ

O OpenVZ é uma solução para paravirtualização utilizando Linux, que trabalha criando ambientes virtuais isolados que funcionam como máquinas convencionais, porém sobre a mesma estrutura de hardware. O OpenVZ cria máquinas virtuais, tornando-as sistemas independentes. Esses ambientes virtuais são conhecidos como VE (*Virtual Environment*) ou VPS (*Virtual Private Server*). No OpenVZ cada VE é tratada independentemente das outras podendo, portanto, possuir *hostname*, acessos de *root*, endereços de rede e demais funcionalidades e características de um sistema não virtualizado [29].

Analisando do ponto de vista das aplicações cada VE é um sistema independente. Esta independência é fornecida pela camada de virtualização do Kernel do *host*. Podemos notar também que somente uma parte dos recursos de processamento do *host* é utilizada pela camada de virtualização do OpenVZ. Outras características da camada de virtualização do OpenVZ e que podemos destacar são [13]:

- Um usuário pode escolher qualquer configuração de arquivos e instalar qualquer programa adicional;
- Uma VE é isolada das demais;
- Processos que pertencem a uma VE são escalonados para execução em qualquer uma das CPU´s disponíveis. Conseqüentemente VE´s não são limitadas somente a uma CPU e podem utilizar toda a capacidade de processamento disponível no momento.

Com relação com a virtualização da rede no OpenVZ podemos destacar:

- Cada VE tem seu próprio endereço IP; porém endereços múltiplos de IP por VE são permitidos;
- O tráfego de rede é isolado das outras VE´s;
- As manipulações da tabela de roteamento e as características avançadas do roteamento são suportadas para cada VE individualmente.

Analisando essas informações a respeito do OpenVZ podemos concluir que é um sistema virtual que trabalha com o conceito de paravirtualização e prima principalmente pelo desempenho.

3 Protocolos de Gerenciamento e Bases de Informações

Um protocolo de gerenciamento deve prover monitoramento e controle de recursos. Atualmente o processo de gerência de recursos computacionais é divido em 5 grandes áreas conforme o padrão de gerenciamento de recursos OSI (*Open Systems Interconnection*). Cada uma destas áreas procura contemplar requisitos diferentes de gerenciamento e são definidos conforme o gerenciamento de falhas, gerenciamento de contabilidade, gerenciamento de configuração, gerenciamento de desempenho e gerenciamento de segurança [14].

Atualmente podemos citar dois protocolos de gerência que norteados por essas cinco áreas funcionais destacam-se no gerenciamento de recursos computacionais. O *Simple Network Management Protocol* (SNMP) e o *Web Based Enterprise Management* (WBEM).

3.1 Protocolo SNMP *(*Simple Network Management Protocol)

O protocolo SNMP foi originalmente desenvolvido para o gerenciamento de redes de computadores, e possui a capacidade de gerenciamento de servidores, roteadores, pontes e demais componentes de uma rede de computadores, onde a tarefa de gerência é realizada, na maioria das vezes, de forma centralizada em alguma estação da rede. Em se tratando do gerenciamento de rede o protocolo de gerência é uma ferramenta importante para auditoria e gerenciamento dos recursos.

O protocolo SNMP é utilizado principalmente para configurar dispositivos remotos, monitorar o desempenho da rede, detectar os acessos inadequados ou falhas, da rede, realizar auditoria sobre o uso da rede, por exemplo [19]. Entretanto, devido a sua flexibilidade também pode ser utilizado para outros tipos de aplicações de gerência.

Ressaltamos também que devido a sua estrutura distribuída e flexível o protocolo SNMP pode ser utilizado para gerência dos mais variados tipos de recursos, incluindo recursos disponíveis em sistemas virtuais, por exemplo.

A estrutura do protocolo SNMP é distribuída e baseada na utilização de gerentes, agentes e em uma base de informações que no caso do SNMP é chamada de *Management Information Base* (MIB).

Os agentes são alocados sobre o sistema que está sendo gerenciado, e o gerente atua realizando solicitações para os agentes. A MIB serve como base de referência de informações para o agente e o gerente definindo uma estrutura hierárquica de acesso aos recursos do sistema. A

Figura 4 apresenta o funcionamento do protocolo SNMP.

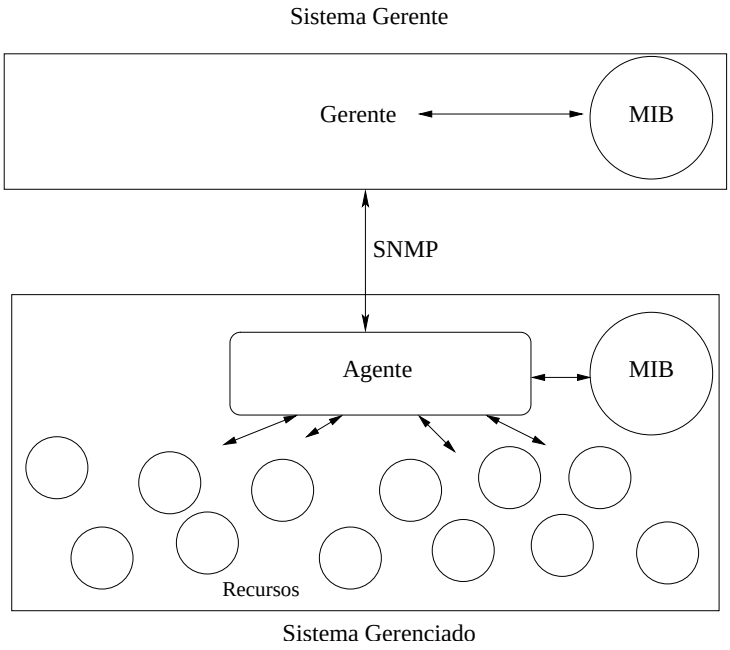

Figura 4 – Funcionamento do Protocolo SNMP.

O funcionamento descrito na Figura 4 apresenta dois dispositivos de uma rede. O primeiro representa um Sistema Gerente que se utiliza de uma MIB como referência e comunica-se com um agente através do protocolo SNMP. O segundo mostra a estrutura de um Sistema Gerenciado, onde podemos observar a utilização da MIB como referência para o agente que acessa informações pertinentes aos recursos dispostos pelo sistema, as informações coletadas pelo agente são repassadas ao gerente utilizando o protocolo SNMP.

O protocolo SNMP destaca-se pela simplicidade de funcionamento, o que se caracteriza pelo reduzido número de operações disponíveis, sendo que todas as operações suportadas pelo SNMP são relacionadas aos objetos de gerência definidos na estrutura da MIB e restringe-se a operações de leitura e escrita. As principais operações disponíveis são as seguintes:

- *Get*: Operação utilizada para a leitura do valor de um objeto, tarefa que caracteriza monitoração;
- *Set*: Operação utilizada para a escrita de um valor em um objeto, tarefa que caracteriza controle;
- *Get-Next*: Operação utilizada para a leitura do valor do próximo objeto;
- *Trap*: Operação utilizada pelo agente para comunicar eventos especiais, tais como problemas de funcionamento do protocolo, por exemplo.

Como observamos o protocolo SNMP implementa gerência de recursos utilizando um número limitado de operações, o que simplifica o processo de gerência, porém acarreta limitações ao protocolo.

É importante ressaltar também que o termo gerência de recursos está relacionado à simbiose entre as operações de monitoração e controle de objetos, sendo portanto, aplicada em ambientes gerenciados onde a MIB proporciona operações de leitura e escrita. Então, uma MIB que proporciona somente operações de leitura caracteriza somente a funcionalidade de monitoração.

No princípio, o gerenciamento de recursos estava diretamente ligado ao gerenciamento de redes, pois as redes de computadores foram o foco inicial da área de gerenciamento de recursos computacionais, pois o fato das redes tornarem-se cada vez maiores, possuírem grande heterogeneidade de recursos, tornarem-se distribuídas, com maior número de protocolos, etc. Devido a esse cenário o protocolo SNMP destaca-se principalmente por ser pioneiro no gerenciamento de redes, e devido a sua facilidade de implantação, utilização, flexibilidade e com naturalidade assumiu o papel de protocolo de gerenciamento dos variados tipos de recursos computacionais.

3.2 Padrão SMI (*Structure Management Information*)

Uma base de informações é um modelo abstrato de referência para o gerenciamento de objetos reais dentro de um sistema computacional, tais como: processador, memória, disco e rede. Atualmente os modelos de protocolos que implementam gerência de rede utilizam bases de informações, dentre eles podemos citar os padrões SNMP que utiliza como base de informações a MIB (*Management Information Base*) e WBEM que utiliza a CIM (*Common Information Model*) explicados anteriormente, portanto uma base de informações bem definida é de suma importância para o gerenciamento adequado de sistemas computacionais [30]. Com esse trabalho foi concebido utilizando o padrão SNMP será dada ênfase neste capítulo ao conceito de bases de informação baseadas no modelo MIB.

A criação de uma base de informações segue regras baseadas nos padrões e normas definidas na SMI (*Structure Management Information*). A estrutura de informações SMI é um grupo de documentos que define [31]:

- Identificação de objetos e de grupos de objetos;
- Sintaxe;
- Tipos de dados reservados a classificar ou quantificar objetos;
- Codificação da informação que será transmitida.

Tais informações descritas são implementadas no desenvolvimento de uma MIB, sendo que atualmente um padrão de MIB que serve como referência para o desenvolvimento das demais é a MIB-II (RFC 1213) [32]. A MIB-II é o aperfeiçoamento natural de sua versão anterior a MIB-I(RFC 1156) [33]. A seção seguinte descreve a MIB-II como referência para o desenvolvimento da vMIB (*Virtual MIB*), que é uma MIB criada para gerenciamento de recursos em máquinas

virtuais, foco desse trabalho, bem como iremos descrever algumas MIB´s desenvolvidas para o Xen e para o VMware a seguir.

O padrão SMI define um objeto a partir de uma seqüência hierárquica de valores, como por exemplo, 1.3.6.1.2.1 que representa a MIB-II. Na Figura 5 podemos observar a MIB-II descrita no formato de árvore, o que facilita a visualização.

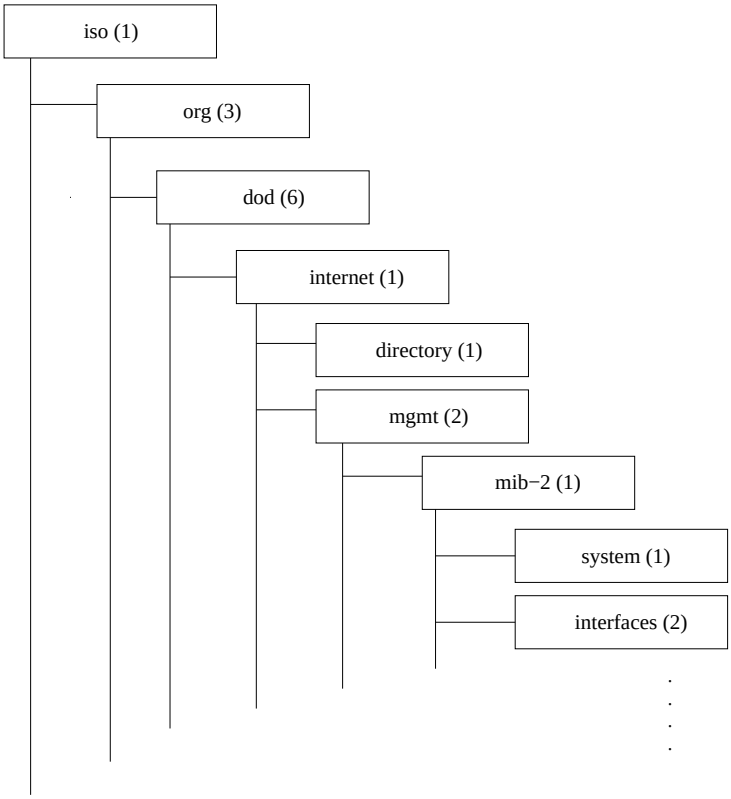

Figura 5 – Estrutura da MIB-II.

Como forma de padronizar objetos, a SMI também realizou a classificação dos tipos de objetos que compõem a MIB. Para tanto, quando se cria uma MIB os seguintes campos devem ser preenchidos: Object Identifier e Object Type, onde o primeiro vai identificar o ramo da árvore ao qual o objeto pertence e o segundo vai definir as características do objeto, sendo composto por mais quatro campos de preenchimento obrigatório.

Os campos que definem as características de cada objeto são: *Syntax* (Sintaxe), *Access* (Acesso), *Status* e *Description* (Descrição), onde cada um deles pode assumir os seguintes valores:

- Syntax : Define o tipo do objeto que pode ser, por exemplo:
  - *Integer* : Um valor inteiro;
  - *DisplayString*: Um valor composto de números e símbolos;
  - *Entry* : Definir o objeto como entrada de valores em uma tabela;
  - *Counter*: Valor inteiro contínuo que somente pode ser incrementado;

  - *Gauge*: Valor inteiro contínuo que pode ser incrementado ou decrementado;
  - *ipaddress*: Endereço de 32 bits no formato de endereçamento IP;
  - *timeticks*: Inteiro, não negativo, que serve para contagem em segundos;
  - *Sequence*: Seqüência de objetos que serão descritos dentro de uma tabela.

- Access : Define o tipo de acesso ao objeto que pode ser, por exemplo:
  - *Read-Only* (RO): Permissão de somente leitura;
  - *Read-Write* (RW): Permissão de leitura e escrita;
  - *Not-Acessible* (NA): Acesso não permitido.

- *Status*: Define o status do objeto que pode ser, por exemplo:
  - *Current*: Estado atual do objeto.

- *Description*: Este campo é reservado para que se descreva o objeto gerenciado.

A seqüência deste capítulo descreve em 3 seções respectivamente a MIB-II, uma MIB para o sistema Xen e uma MIB para o sistema VMware.

A MIB-II é atualmente um padrão de referência para o desenvolvimento de MIB´s. As MIB´s desenvolvidas para gerenciamento de máquinas virtuais Xen e VMware apresentam o atual estado da arte para o gerenciamento de máquinas virtuais.

3.3 MIB-II

Como falamos anteriormente, a MIB-II é uma evolução da MIB-I e acrescenta novos objetos e grupos de objetos a gerência utilizando o padrão SNMP. Hierarquicamente podemos observar sua disposição como objeto na Figura 5, sendo que de modo a exemplificar a sua concepção descreveremos os objetos que compõem a sub-árvore de objetos da MIB-II. Esta estrutura básica da MIB-II pode ser vista na Figura 6.

Na Figura 6, podemos observar os objetos que compõem a MIB-II sendo que podemos decrevê-los da seguinte forma [31], [34], [35], [36], [37], [38]:

- System: Informações sobre o sistema.
- Interface: Informações sobre cada inteface das subredes do sistema.
- AT (*Address Translation*): Este ramo está em desuso, porém realiza o mapeamento da tabela de endereçamento internet para subrede e vice-versa.
- IP (*Intenet Protocol*): Informações relacionadas com o protocolo IP.

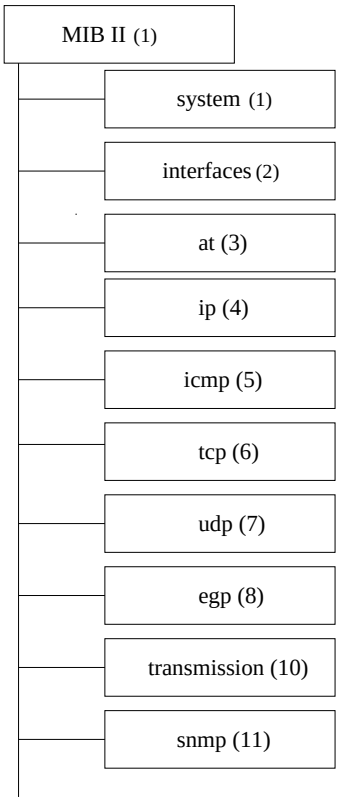

Figura 6 – Sub-árvore da MIB-II.

- ICMP (*Internet Control Message Protocol*): Informações relacionadas com o protocolo ICMP.
- TCP (*Transmission Control Protocol*): Informações relacionadas com o protocolo TCP.
- UDP (*User Datagram Protocol*): Informações relacionadas com o protocolo UDP.
- EGP (*External Gateway Protocol*): Informações relacionadas com o protocolo EGP.
- Dot3: Informações dos esquemas de transmissão e acesso de protocolos sobre cada interface do sistema.
- SNMP: Informações sobre o protocolo SNMP.

Nas seções abaixo, descreveremos as estruturas e funcionalidades de cada uma das camadas que compõem a MIB-II.

3.3.1 Grupo *System*

O grupo *system* provê informações gerais sobre o sistema gerenciado, sendo que para melhor explicarmos tal grupo descreveremos sua estrutura no formato de árvore e descreveremos a

funcionalidade de seus principais objetos. A estrutura do grupo *system* pode ser observada na Figura 7.A Tabela 1 descreve as caracteristicas dos objetos descritos na árvore do grupo system.

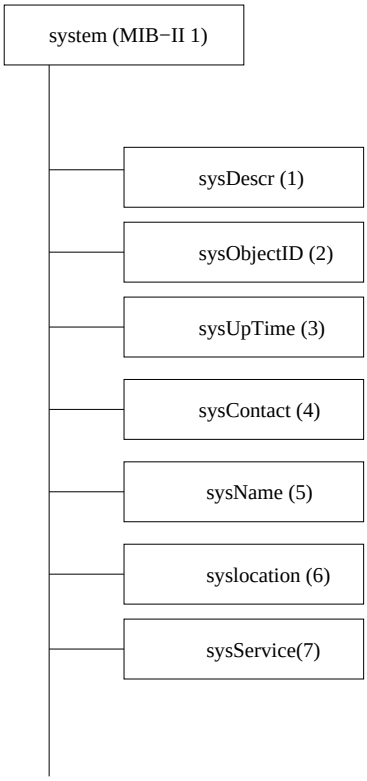

Figura 7 – Sub-árvore do Grupo System.

Tabela 1 – Tabela de descrição dos objetos do grupo *system*.

Objeto	Sintaxe	Acesso	Descrição
sysDescr	DisplayString	RO	Descrição do objeto.
sysObject	Object identifier	RO	ID da empresa desenvolvedora do objeto.
sysUpTime	TimeTicks	RO	Tempo contado (reinicialização).
sysContact	DisplayString	RW	Identificação do contato, gerente da máquina.
sysName	DisplayString	RW	Nome.
sysLocation	DisplayString	RW	Localização física.
sysServices	Integer	RO	Indicação dos serviços primários oferecidos.

3.3.2 Grupo Interfaces

O grupo interfaces provê informações gerais sobre as interfaces físicas do objeto gerenciado, sendo que para melhor explicarmos tal grupo descreveremos sua estrutura no formato de árvore e apresentaremos as funcionalidades de seus principais objetos. Os principais objetos do grupo interfaces podem ser observados na Figura 8.

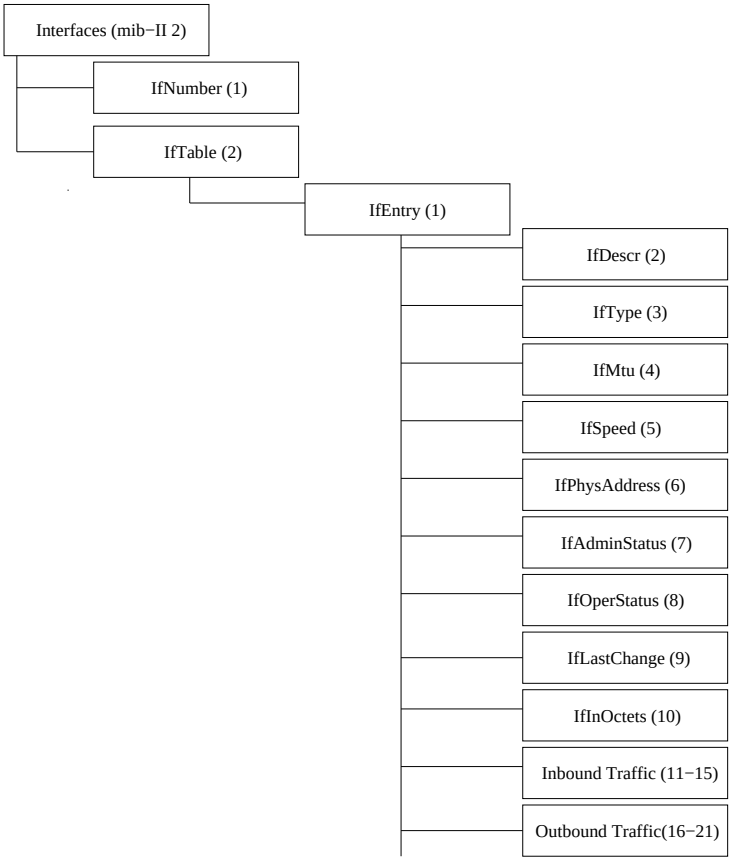

Figura 8 – Sub-árvore do Grupo Interfaces.

A estrutura do grupo interfaces apresenta em seu primeiro nível dois itens: número (*IfNumber*), vai informar o número de interfaces de rede que o sistema possui sem se importar com o seu estado atual, e uma tabela (*IfTable*) recebendo informações de cada uma das intefaces de rede que o sistema possui. Portanto, utilizando a tabela do grupo interfaces para cada interface de rede adicionada uma estrutura hierárquica é criada para monitorá-la, tornando-se mais um objeto a ser gerenciado automaticamente.

Observando a Figura 8, podemos notar que de modo a resumir os objetos da tabela, o primeiro objeto da tabela (IfIndex (1)) não é apresentado na figura, porém destina-se como, qualquer índice da tabela, a definir o valor único e de referência para os valores seguintes da mesma.

Na Figura 8, podemos observar que os objetos de identificação 11 a 15 e 16 a 21 foram resumidos em *inbound Traffic* e *outbound Traffic*, tal ação também visa a reduzir a quantidade de objetos apresentados na Figura 8, visto que o intuito dessa seção é esclarecer como foi concebida a MIB-II e não explicar detalhadamente a mesma. Os objetos do 11 ao 15 representam na ordem *IfUnUcastPkts*, *IfInNUcastPkts*, *IfInDiscards*, *IfInError*s e *IfInUnknownProtos* que representam informações de objetos os quais estão calculando valores relacionados com o tráfego de entrada de dados de cada interface. Os objetos relacionados de 16 a 21 representam na ordem *IfOutOctets*, *IfOutUcastPkts*, *IfOutNUcastPkts*, *ifOutDiscards*,*IfOutErrors* e *IfOutQLen* que representam informações de objetos calculando valores relacionados com o tráfego de

saída de dados de cada interface. Os demais objetos componentes do grupo interfaces podem ser analisados mais sucintamente na Tabela 2.

Tabela 2 – Tabela de descrição dos objetos do grupo interfaces.

Objeto	Sintaxe	Acesso	Descrição
ifDescr	DisplayString	RO	Informações gerais sobre o objeto.
ifType	Integer	RO	Tipo da interface.
ifMtu	Integer	RO	Área de dados do protocolo, em octetos.
ifSpeed	Gauge	RO	Total de dados transportandos pela interface.
ifPhysAddress	PhysAddress	RO	Endereço da interface.
ifAdminStatus	Integer	RW	Alterar status da interface.
ifOperStatus	Integer	RO	Atual status da interface.
ifLastChange	Time ticks	RO	Valor do “sysUpTime” da interface.
ifinOctets	Counter	RO	Total de octetos recebidos pela interface.

3.3.3 Grupo AT (*Address Translation*)

O grupo AT (*Address Translation*) consiste de uma simples tabela, onde cada uma de suas fileiras corresponde a uma interface física do sistema. Porém devido a tal grupo estar obsoleto e não ser utilizado como referência no decorrer do trabalho, o mesmo não será abordado.

3.3.4 Grupo IP (*Internet Protocol*)

O grupo IP (*Internet Protocol*) contém informações sobre a implementação e operação do protocolo IP, desde a utilização do protocolo em sistemas finais (*hosts*) até o seu impacto em sistemas intermediários (*routers*).

O grupo IP contém alguns contadores básicos de tráfego de entrada e saída de dados do protocolo IP. O IP é um dos grupos mais complexos da estrutura da MIB-II, principalmente por ser um grupo que contém vários objetos, dentre os quais podemos destacar a utilização de quatro tabelas *IpAddrTable*, *IpRouteTable*, *IpNetToMediaTable* e *IpForward*.

Sendo que a tabela *IpAddrTable (20)* contém informações sobre o endereçamento IP do sistema, a tabela *IpRouteTable (21)* contém informações sobre o roteamento, o objeto *IpNetToMediaTable (22)* é uma tabela utilizada para realizar o mapeamento de endereços IP com endereços MAC e o objeto *IpForward (24)* é uma tabela nova proposta para substituir a *IpRouteTable* e possui a mesma funcionalidade da anterior, porém possui mais objetos, sendo portanto mais completa.

O grupo IP possui ainda diversos objetos com as mais diferentes funcionalidades, sendo que os principais estão descritos na Figura 9 e podem ser melhor analisados na Tabela 3.

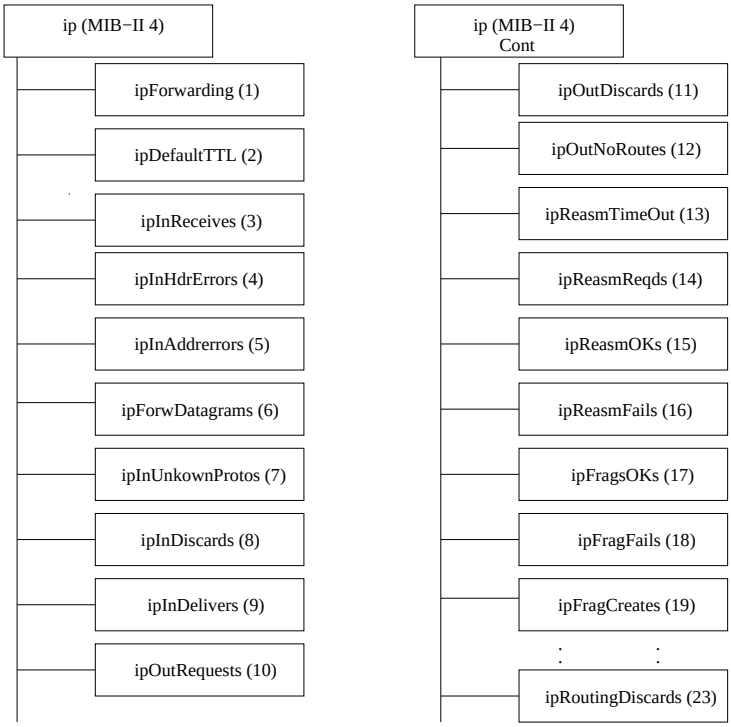

Figura 9 – Sub-árvore do Grupo IP.

Tabela 3 – Tabela de descrição dos objetos do grupo IP.

Objeto	Sintaxe	Acesso	Descrição
ipForwarding	Integer	RW	Pode atuar como gateway(1) ou não (2).
ipDefaultTTL	Integer	RW	Valor *default* do campo *Time-To-Live*.
ipInReceives	Counter	RO	Total de datagramas recebidos.
ipInHdrErrors	Counter	RO	Dat. descartados devido a erros de cabeçalho.
ipInAddrErrors	Counter	RO	Dat. descartados devido a erro de endereçamento.
ipForwDatagrams	Counter	RO	Dat. encaminhados ao gateway.
ipInUnknwnProtos	Counter	RO	Dat. recebidos corretamente,descartados.
ipInDiscard	Counter	RO	Dat. recebidos corretamente,descartados.
ipInDelivers	Counter	RO	Dat. recebidos e entregues corretamente.
ipOutRequests	Counter	RO	Dat. fornecidos para transmissão de IP não local.
ipOutDiscards	Counter	RO	Dat. enviados corretamente,descartados.
ipOutNoRoutes	Counter	RO	Dat. descartados devido a não possuir rota correta.
ipReasmTimeout	Integer	RO	Tempo limite para recebimento de datagramas.
ipReasmReqds	Counter	RO	Dat. recebidos que necessitam ser remontados.
ipReasmOks	Counter	RO	Dat. corretamente remontados.
ipReasmFails	Counter	RO	Dat. que não foram remontados.
ipFragOks	Counter	RO	Dat. corretamente fragmentados.
ipFragFails	Counter	RO	Dat. que não foram fragmentados.
ipFragCreates	Counter	RO	Dat. fragmentados pelo objeto.
ipRoutingDiscards	Counter	RO	Número de rotas inválidas, descartadas.

3.3.5 Grupo ICMP (*Internet Control Message Protocol*)

O grupo ICMP (*Internet Control Message Protocol*) contém informações sobre a implementação e operação do protocolo ICMP. Basicamente o grupo consiste unicamente de *counters* de

vários tipos de mensagens ICMP enviadas e recebidas, como podemos observar na figura 10.

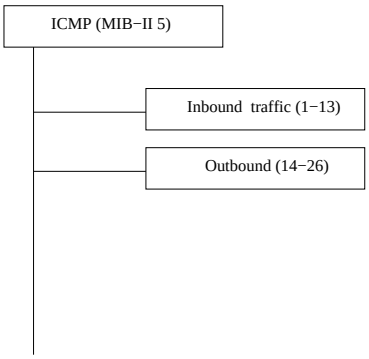

Figura 10 – Sub-árvore do Grupo ICMP.

3.3.6 Grupo TCP (*Transmission Control Protocol*)

O grupo TCP (*Transmission Control Protocol*) contém informações sobre a implementação e operação do protocolo TCP. Os objetos que compõem o grupo estão descritos na Figura 11 e podemos analisar melhor as suas funcionalidades na Tabela 4.

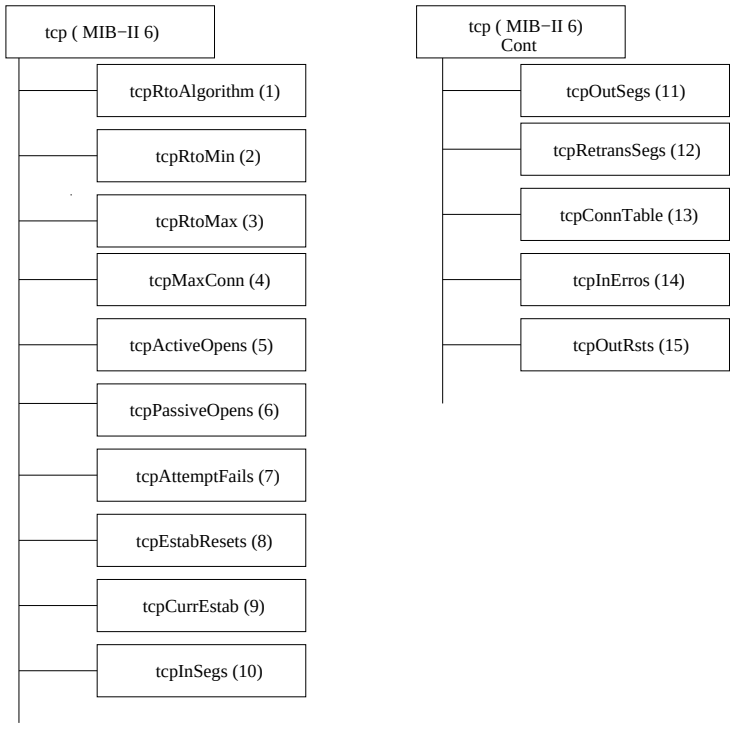

Figura 11 – Sub-árvore do Grupo TCP.

Tabela 4 – Tabela de descrição dos objetos do grupo TCP.

Objeto	Sintaxe	Acesso	Descrição
tcpRtoAlgorithm	Integer	RO	Tempo de retransmissão(Valor).
tcpRtoMin	Integer	RO	Tempo mínimo de retransmissão.
tcpRtoMax	Integer	RO	Tempo máximo de retransmissão.
tcpMaxConn	Integer	RO	Limite de conexões suportadas.
tcpActiveOpens	Counter	RO	Número de conexões ativas suportadas.
tcpPassivesOpens	Counter	RO	Número de conexões passivas suportadas.
tcpAttemptFails	Counter	RO	Número de tentativas de conexões que falharam.
tcpEstabResets	Counter	RO	Número de reinicializações.
tcpCurrEstab	Gauge	RO	Número de conexões *Established* ou *Close-Wait*.
tcpInSegs	Counter	RO	Segmentos recebidos, incluso com erros.
tcpOutSegs	Counter	RO	Segmentos enviados, excluindo os retransmitidos.
tcpRetranSegs	Counter	RO	Segmentos retransmitidos .
tcpConnTable	Sequence	NA	Tabela de conexões TCP.
tcpInErrors	Counter	RO	Segmentos recebidos com erros .
tcpOutRsts	Counter	RO	Segmentos enviados contendo a *RST Flag*.

3.3.7 Grupo UDP (*User Datagram Protocol*)

O grupo UDP (*User Datagram Protocol*) contém informações sobre a implementação e operação do protocolo UDP. Os objetos que compõem o grupo estão descritos na Figura 12 e podemos analisar melhor as suas funcionalidades na Tabela 5.

Um objeto que pode ser destacado na estrutura do grupo UDP é o *udpTable*, tal tabela contém informações sobre os pontos finais do datagrama UPD, tais como uma aplicação local que recebe o conteúdo dos datagramas.

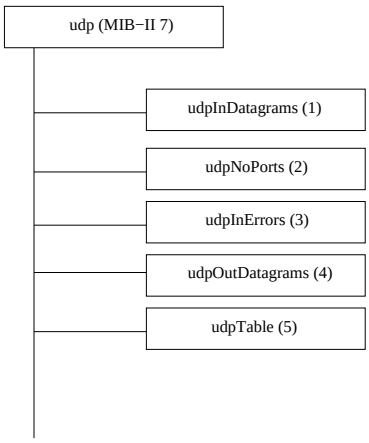

Figura 12 – Sub-árvore do Grupo UDP.

Tabela 5 – Tabela de descrição dos objetos do grupo UDP.

Objeto	Sintaxe	Acesso	Descrição
udpInDatagrams	Counter	RO	Dat. UDP entregues.
udpNoPorts	Counter	RO	Dat. UDP recebidos pela aplicação errada.
udpInErrors	Counter	RO	Dat. UDP recebidos, porém com erros.
udpOutDatagrams	Counter	RO	Dat. UDP enviados.
udpTable	Sequence	NA	Tabela UDP.

3.3.8 Grupo EGP (*External Gateway Protocol*)

O grupo EGP (*External Gateway Protocol*) contém informações sobre a implementação e operação do protocolo EGP. Incluindo informações sobre as mensagens EGP enviadas e recebidas. Para melhor explicarmos o grupo, destacamos os principais objetos na Figura 13 e podemos analisar sucintamente suas funcionalidades na Tabela 6.

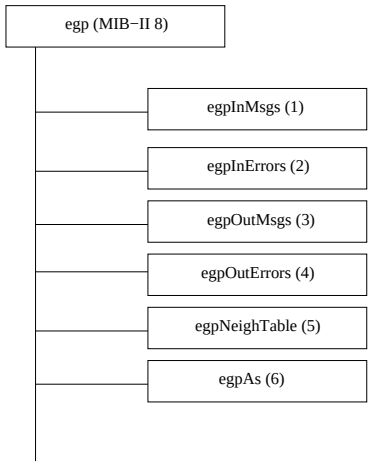

Figura 13 – Sub-árvore do Grupo EGP.

Tabela 6 – Tabela de descrição dos objetos do grupo EGP.

Objeto	Sintaxe	Acesso	Descrição
egpInMsgs	Counter	RO	Mensagens EGP recebidas sem erros.
egpInErrors	Counter	RO	Mensagens EGP recebidas com erro.
egpOutMsgs	Counter	RO	Mensagens EGP enviadas.
egpOutErrors	Counter	RO	Mensagens EGP não enviadas, falha de protocolo.
egpNeightTable	Sequence	NA	Tabelas EGP.
egpAs	Integer	RO	Número de sistemas.

3.3.9 Grupos Dot3 e SNMP

Os protocolos Dot3 e SNMP, terminam por compor o 1º nível da estrutura da MIB-II, porém por não serem significantes dentro do contexto do trabalho, não serão apresentados.

3.4 Grupo *Host Resources MIB*

O Grupo *Host Resources* MIB pertence a MIB-II e está definido conforme a estrutura hierárquica 1.3.6.1.2.1.25 . Este grupo define informações a respeito de sistemas que funcionam como hospedeiros de máquinas virtuais [39]. Os objetos que compõem o grupo estão descritos na Figura 14 e podemos analisar melhor as suas funcionalidades na Tabela 7.

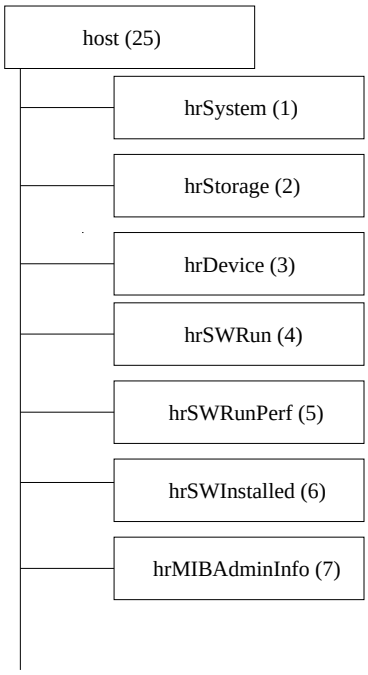

Figura 14 – Sub-árvore do Grupo *Host Resources MIB*.

Os primeiros três objetos que compõem o grupo *host* definem informações sobre os recursos disponíveis, e os demais apresentam informações sobre características do software. Por serem mais importantes no desenvolvimento do trabalho, os três primeiros objetos serão descritos a seguir.

O grupo *hrSystem* é composto por objetos que definem informações a respeito do sistema que hospeda máquinas virtuais. Os objetos que compõem o grupo estão descritos na Figura 15 e podemos analisar melhor as suas funcionalidades na Tabela 8.

O grupo *hrStorage* é composto por objetos que definem informações a respeito de dispositivos de armazenamento e memória do sistema que hospeda máquinas virtuais. Os objetos que

Tabela 7 – Tabela de descrição dos objetos do grupo *Host Resources*.

Objeto	Sintaxe	Acesso	Descrição
hrSystem	DisplayString	RO	Sistema.
hrStorage	DisplayString	RO	Armazenamento e memória.
hrDevice	DisplayString	RO	Dispositivos de rede, CPU, etc.
hrSWRun	DisplayString	RO	Informações sobre o software.
hrSWRunPerf	DisplayString	RO	Desempenho do sistema.
hrSWInstalled	DisplayString	RO	Atualizações de software.
hrMIBAdminInfo	DisplayString	RO	Informações Gerais.

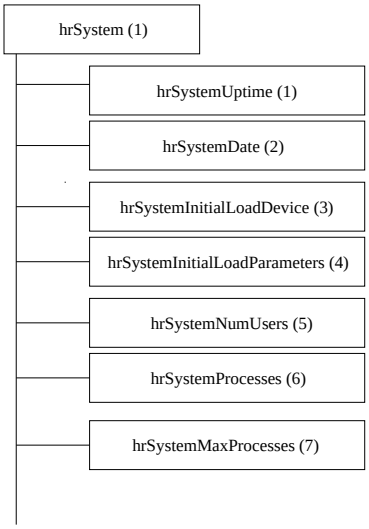

Figura 15 – Sub-árvore do Grupo *hrSystem*.

Tabela 8 – Tabela de descrição dos objetos do grupo *hrSystem*.

Objeto	Sintaxe	Acesso	Descrição
hrSystemUptime	Time Ticks	RO	Tempo contado (reinicialização).
hrSystemDate	DisplayString	RW	Data e Tempo.
hrSystemInitialLoadDevice	Integer	RW	Configuração inicial.
hrSystemInitialLoadParameters	DisplayString	RW	Parâmetros da Conf inicial.
hrSystemNumUsers	Gauge	RO	Número de sessões de usuário.
hrSystemProcesses	Gauge	RO	Processos Carregados.
hrSystemMaxProcesses	Integer	RO	Número de processos suportados.

compõem o grupo estão descritos na Figura 16 e podemos analisar melhor as suas funcionalidades na Tabela 9.

O grupo *hrDevice* é composto por objetos que definem informações a respeito de dispositivos compondo os recursos do sistema que hospeda máquinas virtuais. Os objetos que compõem o grupo são basicamente tabelas as quais descrevem informações pertinentes a cada um dos recursos disponíveis e estão descritos na Figura 17.

Como podemos observar na 17, o grupo *hrDevice* é composto, em sua maioria, por tabelas,

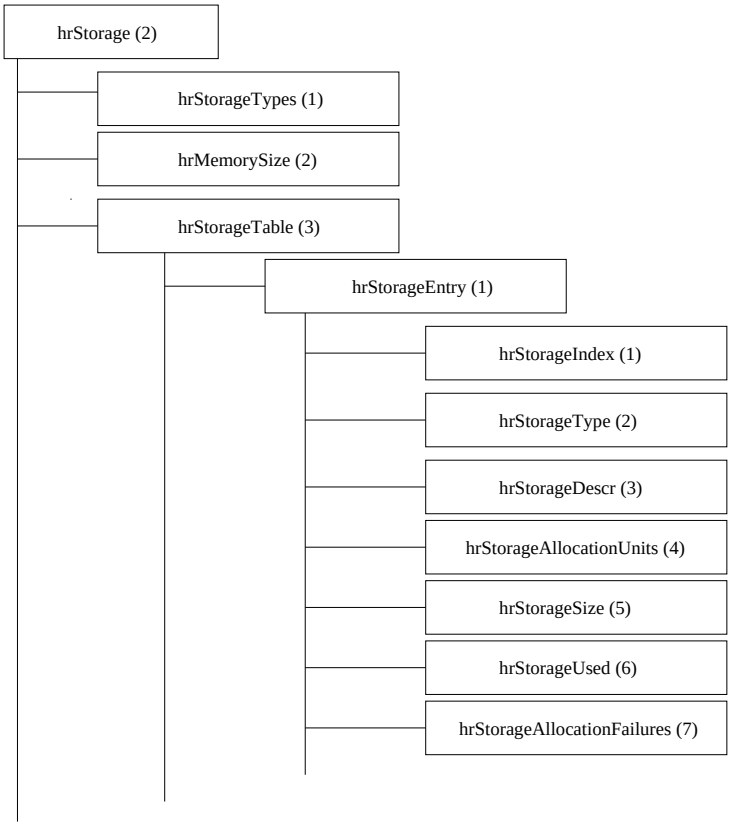

Figura 16 – Sub-árvore do Grupo *hrStorage*.

Tabela 9 – Tabela de descrição dos objetos do grupo *hrStorage*.

Objeto	Sintaxe	Acesso	Descrição
hrStorageTypes	DisplayString	RO	Dispositivos Aceitos.
hrMemorySize	Integer	RO	Memória RAM disponível.
hrStorageTable	Sequence	NA	Tabela.
hrStorageEntry	storageEntry	NA	Entrada da Tabela.
hrStorageIndex	Integer	RO	Índice da tabela.
hrStorageType	DisplayString	RO	Dispositivo utilizado.
hrStorageDescr	DisplayString	RO	Tipo e instância do dispositivo.
hrStorageAllocationUnits	Integer	RO	Tamanho(KB) do dispositivo.
hrStorageSize	Integer	RW	Alterar o objeto anterior.
hrStorageUsed	Integer	RO	Quantidade de dispositivos.
hrStorageAllocationFailures	Counter	RO	Número de falhas.

dentre as quais podemos destacar *hrProcessorTable*, *hrNetworkTable* e *hrDiskStorageTable* que representam respectivamente informações sobre processador, rede e disco.

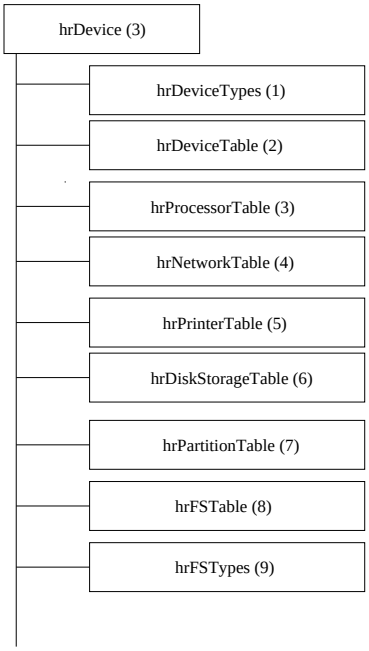

Figura 17 – Sub-árvore do Grupo *hrDevice*.

3.5 Xen MIB

Esta seção visa a apresentar uma MIB para o gerenciamento de máquinas virtuais Xen de modo a apresentar o estado atual da tecnologia de gerenciamento de sistemas virtuais.

Uma MIB criada para gerenciar máquinas virtuais Xen foi desenvolvida por um grupo de pesquisa da Universidade de *Braunschweig*, Alemanha. Essa MIB descreve máquinas virtuais Xen e os dispositivos físicos aos quais o sistema Xen está alocado, habilitando assim o monitoramento dos *hosts* e das máquinas virtuais [40].

A definição dessa MIB é simples e provê monitoração efetiva de objetos. Cabe ressaltar, porém, que utilizamos o termo monitoração para nos referimos à Xen MIB, pois a mesma possui apenas objetos *read-only*, permitindo apenas a opção de leitura dos objetos. Logo, podemos afirmar que a mesma não oferece opção de escrita (*read-write*), não sendo portanto cabível ao termo gerenciamento.

A Figura 18 descreve o 1ºnível da estrutura da Xen MIB, sendo que o mesmo é composto por 3 objetos, sendo que o primeiro (*xenObjects*) define os recursos que são monitorados pela MIB e os demais definem propriedades da MIB. Explicaremos sucintamente o primeiro grupo dando ênfase para os objetos que compõe a estrutura.

A Figura 19 mostra a estrutura do 1º nível de objetos que compõe o grupo xenObjects, onde podemos analisar a divisão dos recursos monitorados em quatros grandes grupos *xenHost*, *xenDomainTable*, *xenVcpuTable* e *xenNetworkTable* dispostos de acordo como a funcionalidade de cada objeto. A Tabela 10 descreve os objetos.

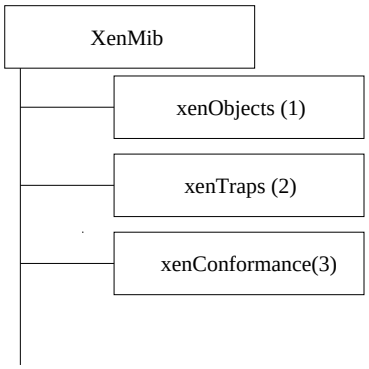

Figura 18 – Estrutura da Xen MIB.

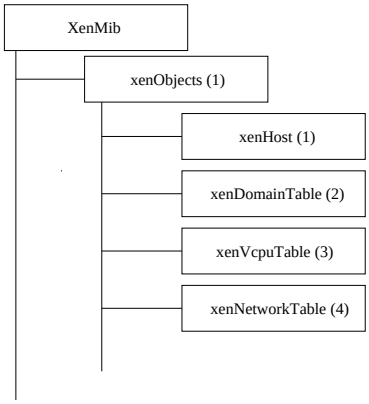

Figura 19 – Sub-árvore Xen Objects da Xen MIB.

Tabela 10 – Tabela de descrição dos objetos do grupo xenObjects.

Objeto	Sintaxe	Acesso	Descrição
xenHost			Descrição dos Objetos do Host .
xenDomainTable	Sequence	NA	Lista de todos os domínios Xen.
xenVcpuTable	Sequence	NA	Lista de todas as Vcpus por domínio.
xenNetworkTable	Sequence	NA	Lista de todas as redes por domínio.

O grupo *xenHost* é composto por objetos os quais definem as características gerais do *host* hospedando as máquinas virtuais. Especificamente através dele podemos obter informações a respeito da versão do Xen que suporta as máquinas virtuais e sobre a quantidade de CPU´s virtuais que compõem o sistema, por exemplo. O grupo *xenHost* pode ser analisado no formato de árvore através da Figura 20, e a descrição dos objetos que compõe este grupo pode ser observada na Tabela 11.

O grupo *xenDomainTable* é uma tabela composta por objetos definindo as características do domínio que gerencia as máquinas virtuais. Através dele podemos obter informações a respeito do nome, estado e quantidade de memória disponível para o domínio, por exemplo. O grupo *xenDomainTable* pode ser analisado no formato de árvore através da Figura 21 e a descrição

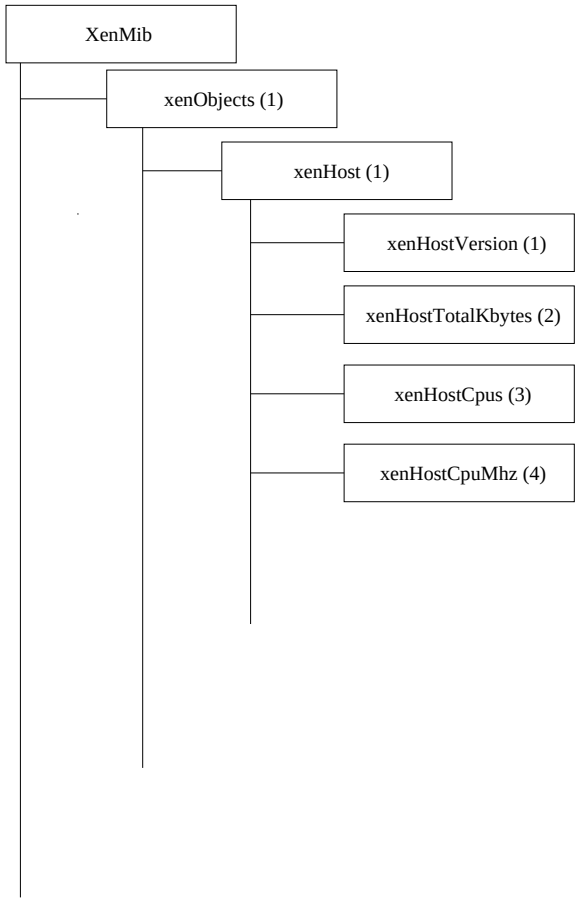

Figura 20 – Sub-árvore Host da Xen Objects.

Tabela 11 – Tabela de descrição dos objetos do grupo xenHost.

Objeto	Sintaxe	Acesso	Descrição
xenHostXenVersion	SnmpAdminString	RO	Versão do Xen em execução .
xenHostTotalMemKBytes	Unsigned32	RO	Memória disponível(Kbytes).
xenHostCpus	Unsigned32	RO	CPU´s do host físico.
xenHostCpuMhz	Unsigned32	RO	Freqüencia em Mhz das Cpus.

dos objetos que compõe este grupo pode ser observada na Tabela 12.

Tabela 12 – Tabela de descrição dos objetos do grupo xenDomainTable.

Objeto	Sintaxe	Acesso	Descrição
xenDomainEntry	xenDomainEntry	NA	Entrada da tabela.
xenDomainName	SnmpAdminString	NA	Nome do domínio xen.
xenDomainState	xenDomainState	RO	Estado do domínio xen.
xenDomainMemKBytes	Unsigned32	RO	Memória ocupada.
xenDomainMaxMemKBytes	Unsigned32	RO	Memória disponível.

O grupo *xenVcpuTable* é uma tabela composta por objetos que definem as características das CPU´s virtuais as quais estão presentes nas máquinas virtuais. O grupo *xenVcpuTable* pode ser analisado no formato de árvore através da Figura 22, e a descrição dos objetos desse grupo

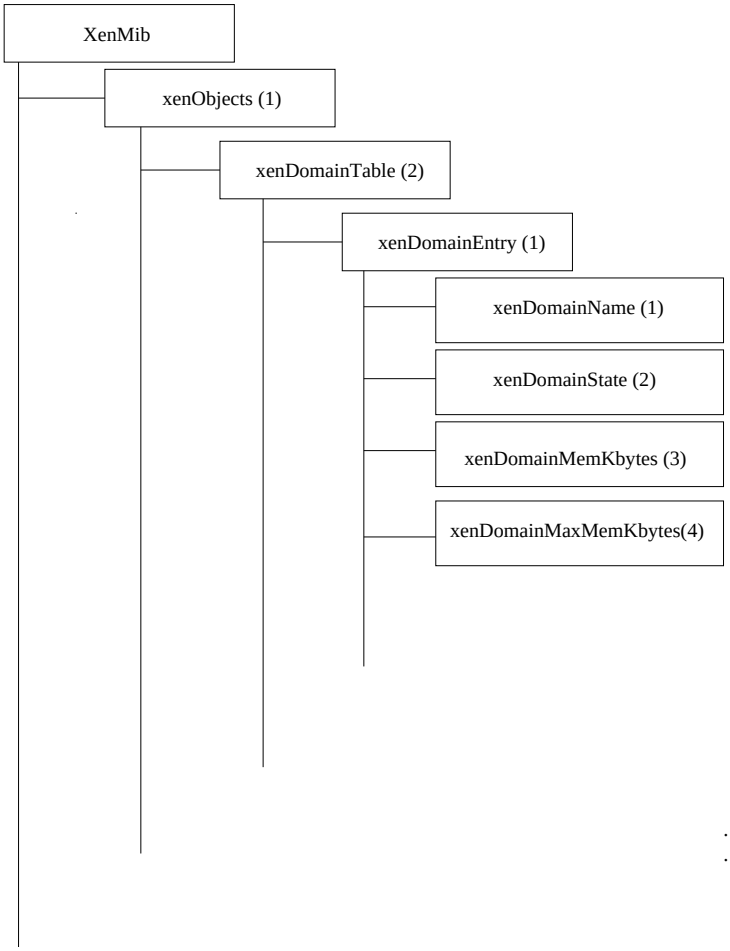

Figura 21 – Sub-árvore Domain da Xen Objects.

pode ser observada na Tabela 13.

Tabela 13 – Tabela de descrição dos objetos do grupo xenVcpuTable.

Objeto	Sintaxe	Acesso	Descrição
xenVcpuEntry	xenVcpuEntry	NA	Entrada da tabela.
xenVcpuIndex	Unsigned32	NA	Index da tabela Vcpu.
xenVcpuMilliseconds	Counter	RO	Millisegundos consumidos pela Vcpu.

O grupo *xenNetworkTable* é uma tabela composta por objetos que definem as características da rede de comunicação das máquinas virtuais. Através dele podemos obter informações gerais a respeito das interfaces de rede, tais como quantidade de kbytes e pacotes recebidos ou enviados, por exemplo. O grupo *xenNetworkTable* pode ser analisado no formato de árvore através da Figura 23, e a descrição dos objetos que compõe este grupo pode ser observada na Tabela 14.

Esta MIB desenvolvida para o sistema virtual Xen proporciona monitoração em objetos que compõem o sistema Xen, sendo que de modo geral é dividida em grupos definindo informações gerais sobre o sistema e informações mais específicas sobre os domínios, CPU´s e dispositivos de rede.

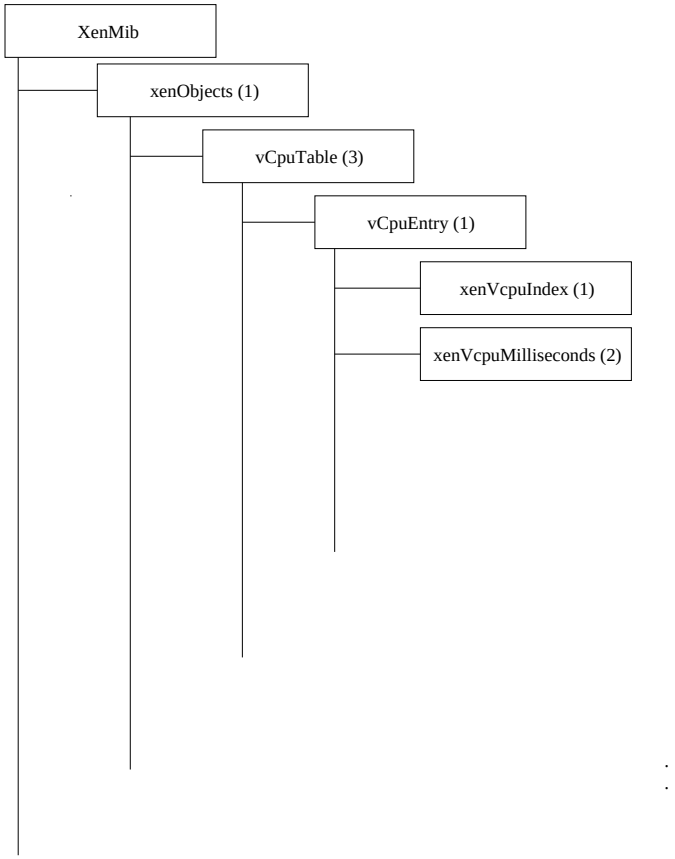

Figura 22 – Sub-árvore VCPU da Xen Objects.

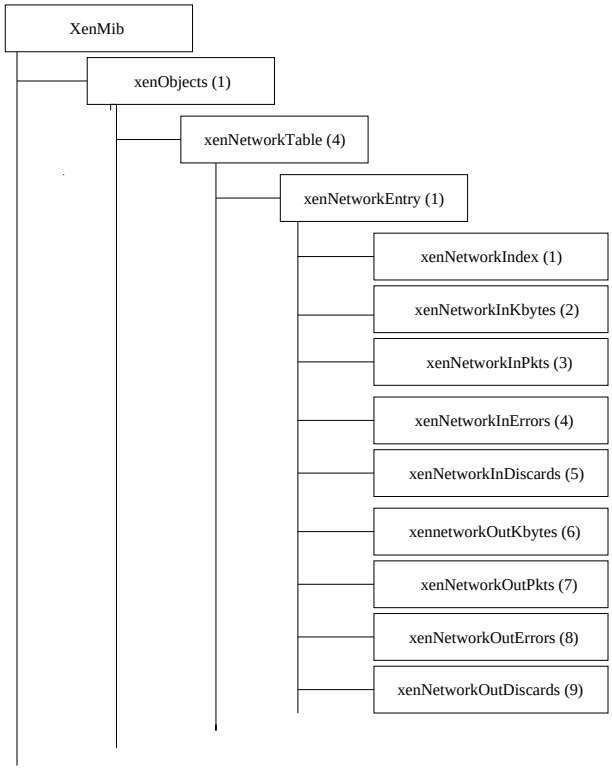

Figura 23 – Sub-árvore Network da Xen Objects.

Tabela 14 – Tabela de descrição dos objetos do grupo xenNetworkTable.

Objeto	Sintaxe	Acesso	Descrição
xenNetworkEntry	xenNetworkEntry	NA	Entrada da tabela.
xenNetworkIndex	Unsigned32	NA	Index da tabela rede.
xenNetworkInKBytes	Counter	RO	KBytes recebidos.
xenNetworkInPkts	Counter	RO	Pacotes recebidos.
xenNetworkInErrors	Counter	RO	Pacotes com erros recebidos.
xenNetworkInDiscards	Counter	RO	Pacotes não recebidos.
xenNetworkOutKBytes	Counter	RO	KBytes enviados.
xenNetworkOutPkts	Counter	RO	Pacotes enviados.
xenNetworkOutErrors	Counter	RO	Pacotes não enviados.
xenNetworkOutDiscards	Counter	RO	Pacotes descartados.

3.6 VMware MIB

Esta seção apresenta um modelo de MIB para o gerenciamento de máquinas virtuais VMware com o intuito de analisarmos uma base de informações desenvolvida para tal sistema virtual.

A VMWARE-ROOT-MIB foi desenvolvida com a finalidade de monitorar máquinas virtuais hospedadas sobre o sistema VMware ESX. Essa MIB descreve de maneira hierárquica e bem definida os objetos reais e virtuais [41].

A definição da MIB é simples e provê monitoração efetiva dos objetos, porém podemos notar que semelhantemente a descrição da Xen MIB a VMWARE-ROOT-MIB realiza apenas a monitoração dos objetos que a compõem, visto que a mesma possui apenas objetos *read-only*, permitindo apenas a opção de leitura dos objetos. Portanto podemos afirmar que a VMWARE-ROOT-MIB não oferece opção de escrita (*read-write*), não sendo portanto possível implementar o gerenciamento através dela.

A Figura 24 descreve o objeto *VMware-Resources* da estrutura da VMWARE-ROOT-MIB. O *VMware-Resources* é o um objeto composto pelos recursos do sistema divididos em quatro grupos *vmwResources 1*, *vmwResources 2*, *vmwResources 3* e *vmwResources 4*. Explicaremos a seguir cada um destes grupos descrevendo seus objetos.

O grupo *vmwResources 1* descreve os objetos relacionados ao recurso CPU do VMware-ESX. Através dele pode-se obter informações sobre a quantidade de CPU´s físicas e virtuais, por exemplo. A Figura 25 apresenta o grupo *vmwResources 1* no formato de árvore a descrição dos objetos que compõe este grupo pode ser observada na Tabela 15.

O grupo *vmwResources 2* descreve os objetos relacionados com os recursos de memória do VMware-ESX. Através dele pode-se obter informações sobre a quantidade de memória utilizada e disponível, por exemplo. A Figura 26 apresenta o grupo *vmwResources 2*, no formato de árvore, a descrição dos objetos que compõe este grupo pode ser observada na Tabela 16.

O grupo *vmwResources 3* descreve os objetos relacionados com o recurso de disco do VMware-ESX. Através dele, pode-se obter informações sobre as quantidades de leituras e es-

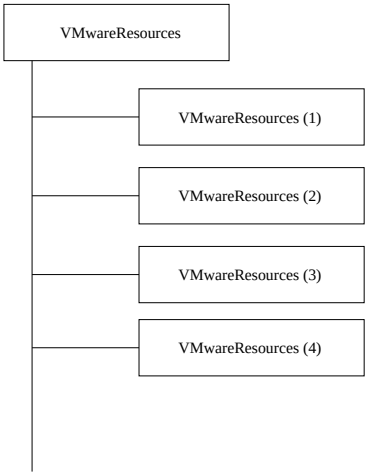

Figura 24 – Estrutura da Vmware MIB.

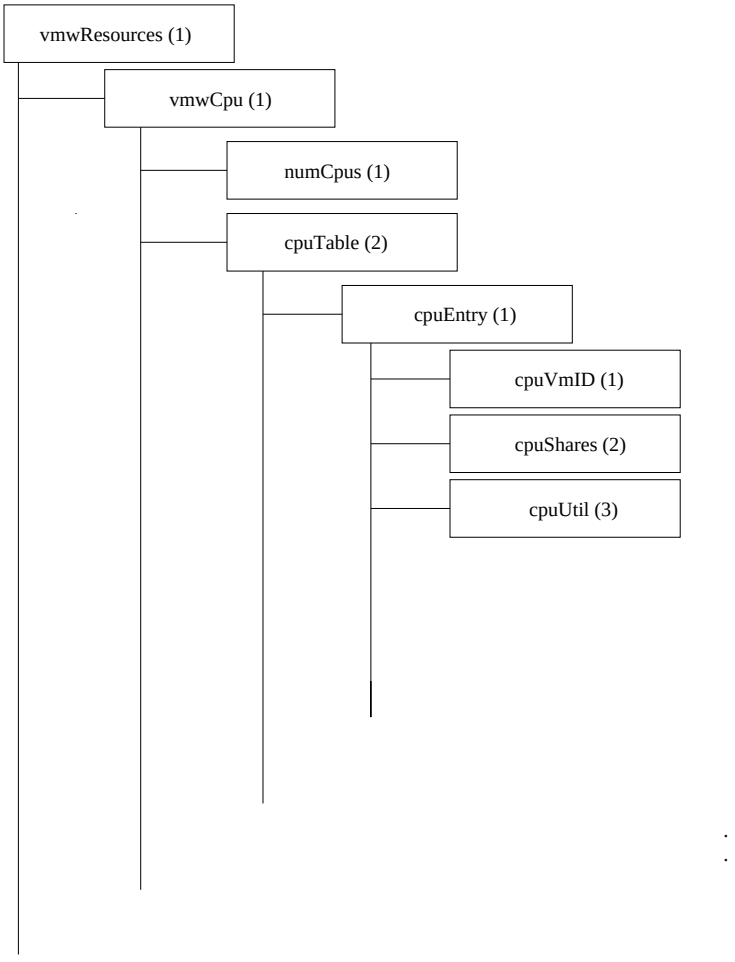

Figura 25 – Sub-árvore da Vmware Resources 1.

critas realizadas nos discos, por exemplo. A Figura 27 apresenta o grupo *vmwResources 3*, no formato de árvore, a descrição dos objetos que compõe este grupo pode ser observada na Tabela 17.

O grupo *vmwResources 4* descreve os objetos relacionados com os recursos de rede do

Tabela 15 – Tabela de descrição dos objetos do grupo vmwResources 1.

Objeto	Sintaxe	Acesso	Descrição
vmwCpu			Item único do grupo vmwResources 1.
numCpus	Integer	RO	Quantidade de CPU´s físicas do sistema.
cpuTable	Sequence	NA	Tabela de CPU´s para máquinas virtuais.
cpuEntry	CpuEntry	NA	Entrada da tabela de CPU´s para máquinas virtuais.
cpuVMID	Integer	RO	ID da máquina virtual.
cpuShares	Integer	RO	CPU´s alocadas e compartilhadas na máquina virtual.
cpuUtil	Integer	RO	Tempo de funcionamento da máquina virtual.

Tabela 16 – Tabela de descrição dos objetos do grupo vmwResources 2.

Object	Syntax	Access	Description
vmwMemory			Item único do grupo vmwResources 2.
memSize	Integer	RO	Memória física da máquina.
memCOS	Integer	RO	Memória física usada pelo console.
memAvail	Integer	RO	Memória física disponível.
memTable	Sequence	NA	Tabela para memória das máquinas virtuais.
memEntry	memEntry	NA	Entrada da tabela de memória.
memVMID	Integer	RO	ID da máquina virtual.
memShares	Integer	RO	Memória alocada na máquina virtual.
memConfigured	Integer	RO	Memória configurada na máquina virtual.
memUtil	Integer	RO	Memória utilizada pela máquina virtual.

Tabela 17 – Tabela de descrição dos objetos do grupo vmwResources 3.

Objeto	Sintaxe	Acesso	Descrição
vmwHBATable	Sequence	NA	Tabela de disco das máquinas virtuais.
hbaEntry	hbaEntry	NA	Entrada da tabela de disco.
hbaIdx	Integer	RO	Index da tabela hba.
hbaName	DisplayString	RO	Descrição do disco.
hbaVMID	Integer	RO	ID da máquina virtual.
diskShares	Integer	RO	Disco alocado na máquina virtual.
numRead	Integer	RO	Quantidade de leituras no disco.
numWrites	Integer	RO	Quantidade de escritas no disco.
kbWritten	Integer	RO	Quantidade de escritas no disco(KB).

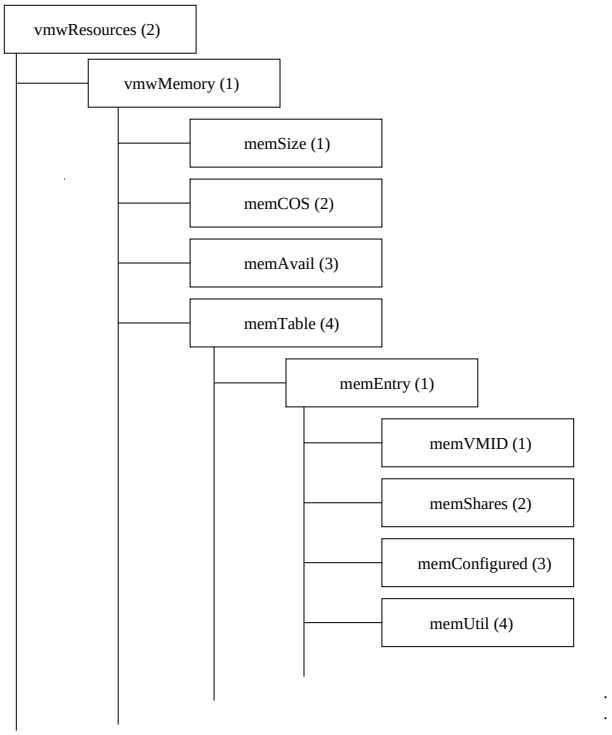

Figura 26 – Sub-árvore da Vmware Resources 2.

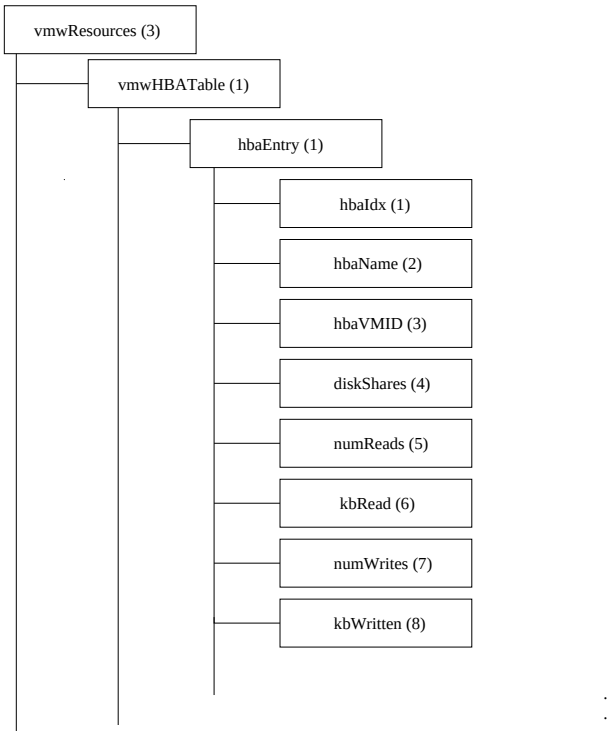

Figura 27 – Sub-árvore da Vmware Resources 3.

VMware-ESX. Através dele pode-se obter informações sobre os nomes das intefaces de rede e as quantidades de pacotes recebidos e enviados por cada interface, por exemplo. A Figura 28 apresenta o grupo *vmwResources 4*, no formato de árvore, a descrição dos objetos que compõe

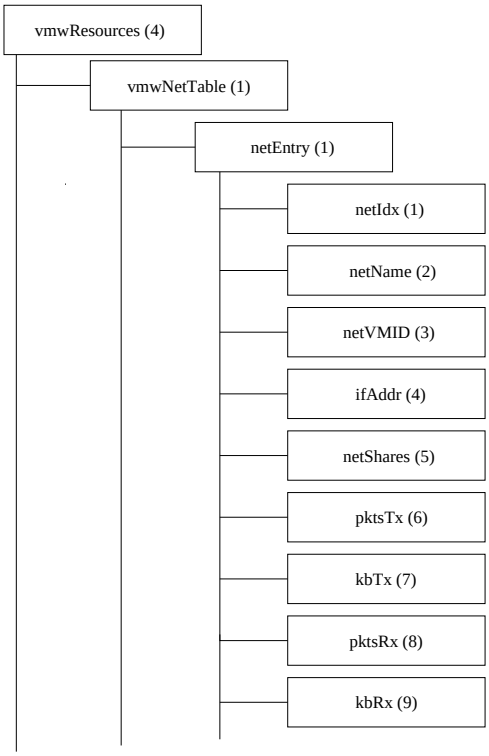

Figura 28 – Sub-árvore da Vmware Resources 4.

Tabela 18 – Tabela de descrição dos objetos do grupo vmwResources 4.

Objeto	Sintaxe	Acesso	Descrição
vmwNetTable	Sequence	NA	Tabela para rede das máquinas virtuais.
vmwEntry	vmwEntry	NA	Entrada da tabela de rede.
netIdx	Integer	RO	Index da tabela rede.
netName	DisplayString	RO	Descrição da interface de rede.
netVMID	Integer	RO	ID da máquina virtual.
ifAddr	Display-String	RO	Endereço MAC da máquina virtual.
netShares	Integer	NA	Compartilhamento da banda de rede.
pktsTx	Integer	RO	Pacotes transmitidos pela interface de rede.
kbTx	Integer	RO	Kbytes transmitidos pela interface de rede.
pktsRx	Integer	RO	Pacotes recebidos pela interface de rede.
kbRx	Integer	RO	Kbytes recebidos pela interface de rede.

este grupo pode ser observada na Tabela 18.

A MIB VMWARE-ROOT-MIB proporciona monitoração em objetos que compõem o sistema VMware ESX, sendo que de modo geral é dividida em grupos que definem informações gerais e específicas sobre objetos físico e virtuais que definem os recursos de processamento, memória, disco e rede.

Neste capítulo, analisamos os protocolos de gerenciamento e bases de informações que podem ser utilizadas como ferramentas para o gerenciamento de sistemas virtuais, atualmente podemos observar a tendência de pesquisas relacionadas ao gerenciamento de sistemas virtuais,

seja no desenvolvimento de ferramentas, na adoção de técnicas ou na utilização de tecnologias já estabelecidas como no caso do protocolo SNMP e suas bases de informações como a *Host Resources*, a Xen MIB e a VMware MIB tratadas acima. O que podemos concluir observando essas tecnologias e tendências é que talvez o mais importante seja a definição de regras e padrões para o gerenciamento de sistemas virtuais [42]. Neste cenário a utilização adequada de protocolos e bases de informações pode ser de grande valia no processo de desenvolvimento de técnicas e padrões de gerenciamento para sistemas virtuais.

4 Desenvolvimento da Virtual MIB (vMIB)

Como observamos nos capítulos sobre a Xen MIB e a VMware MIB, atualmente estão a disposição MIB´s para o gerenciamento de máquinas virtuais distintas, ou seja, se possui uma MIB para gerência de máquinas virtuais especificamente deste ou daquele sistema virtual.

Quando trabalhamos com máquinas virtuais, visamos a os mais variados tipos de aplicações, sendo que dependendo do tipo de aplicação este ou aquele sistema virtual que é mais adequado para tal, não é necessariamente o mais indicado para um outro tipo de aplicação.

Neste cenário, uma MIB genérica para o gerenciamento de máquinas virtuais é uma ferramenta importante, pois torna padronizada e flexível para a tarefa de gerência deste tipo de ambiente que se caracteriza por ser distribuído e heterogêneo.

Como obsevamos anteriormente, atualmente existem MIB´s que trabalham com sistemas virtuais. Porém, uma grande deficiência é notada em ambas: a incapacidade de controlar objetos utilizando o protocolo SNMP. Essa incapacidade é decorrente do desenvolvimento das mesmas, visto que nenhuma possui objetos que possam ser alterados (*read-write*) o que caracteriza apenas atividades de monitoração.

A Virtual MIB (vMIB) foi concebida com o intuito de prover gerenciamento de máquinas virtuais independentemente do sistema virtual utilizado e também possui a finalidade de viabilizar a interoperabilidade no gerenciamento entre sistemas virtuais diferentes.

4.1 Características

A vMIB foi criada segundo os padrões da SMI, baseada na MIB-II e nos modelos desenvolvidos para o Xen e para o VMware ESX. A vMIB possui como principais funcionalidades a interoperabilidade no gerenciamento de sistemas virtuais diferentes e a capacidade de gerenciar objetos físicos e virtuais.

A vMIB possui algumas características importantes que a diferencia das MIB´s desenvolvidas para os sistemas virtuais Xen e VMware, dentre elas destaca-se a capacidade de prover a seus agentes operações de leitura e escrita, caracterizando as atividades de monitoração e controle, proporcionando, portanto, a atividade de gerência de recursos.

Outra funcionalidade inovadora da vMIB é a sua capacidade de interoperar a atividade de gerência entre diversos sistemas virtuais, ou seja, através do mesmo protocolo e base de informações podemos obter informações de sistemas virtuais diferentes, como por exemplo o

Xen, o VMware e o OpenVZ, ou até mesmo gerenciar um grupo de máquinas virtuais com diversos sistemas virtuais diferentes em funcionamento simultâneo em um ambiente como um *cluster* [43] ou uma grade [44], por exemplo. Tal característica agrega flexibilidade à tarefa de gerenciamento dos sistemas virtuais, dos *hosts* e das máquinas virtuais. Cabe ressaltar que para o funcionamento desse tipo de estrutura faz-se necessário o desenvolvimento de agentes distintos para cada tipo de sistema virtual.

A Figura 29 apresenta o funcionamento do protocolo SNMP para gerência de máquinas virtuais utilizando a vMIB com agentes Xen e VMware comunicando-se com uma estação gerente. Podemos observar, na Figura, os agentes utilizando a vMIB como referência ao acesso e obtenção de informações pertinentes aos recursos de cada sistema virtual, e a centralização da tarefa de gerência através da comunicação dos agentes com o sistema gerente utilizando o protocolo SNMP.

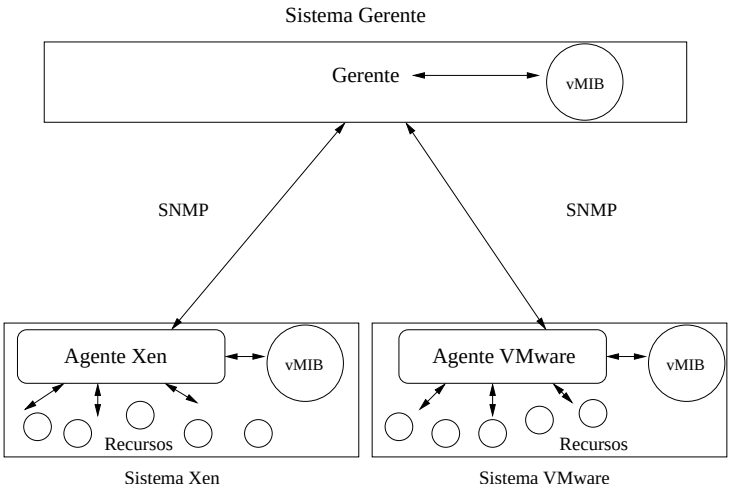

Figura 29 – Interoperabilidade utilizando a vMIB.

A utilização de uma MIB genérica também acarreta em dificuldades no gerenciamento dos sistemas virtuais específicos aos quais ela se propõe a gerenciar, visto que recursos peculiares a este ou aquele sistema não são considerados, pois a vMIB visa, principalmente, a atender a demanda de recursos que se encontram em qualquer tipo de sistema virtual.

Outro fator que termina contribuindo para esse tipo de deficiência é o fato de que existem atualmente diversos sistemas virtuais, versões e técnicas de implementar a virtualização, assim cada sistema virtual pode implementar a virtualização utilizando uma técnica diferente, sendo que um mesmo sistema pode alterar essa técnica de uma versão para outra visando a adaptar-se aos diversos tipos de aplicações que se utilizam de sistemas virtuais.

Podemos nesse contexto citar o VWware, que talvez venha a ser o sistema virtual que possui o maior número de versões. O VMware possui, em sua maioria, versões que implementam a virtualização através da técnica de emulação, porém encontramos versões do mesmo que implementam a virtualização através de paravirtualização, podemos vislumbrar, que: se uma MIB específica para o sistema VMware fosse elaborada a mesma também encontraria problemas,

pois a técnica utilizada para implementar a virtualização tem impacto direto na quantidade e tipo de recurso gerenciado.

Uma MIB genérica torna-se importante, pois padroniza a gerência de recursos em sistemas virtuais, e possibilita a seus usuários atividades de monitoração e controle sobre os principais recursos que compõe um sistema virtual através de uma mesma base de informações, mesmo que como podemos notar a generalização no desenvolvimento desse tipo de MIB faz com que alguns recursos não sejam considerados em um ou outro sistema virtual.

A vMIB segue alguns padrões adotados nas MIB´s que trabalham com os sistemas Xen e VMware-ESX. Esses padrões poderão ser melhor visualizados posteriormente quando observarmos a estrutura de disposição e a funcionalidade dos objetos que compõem a vMIB. Podemos notar também objetos com funcionalidades descritas nas outras MIB´s, bem como novos objetos e em alguns casos mudanças na disposição de objetos.A Figura 30 mostra a disposição da vMIB dentro da estrutura hierárquica da definição SMI, permitindo que a mesma seja representada pela seqüência 1.3.6.1.3.1.

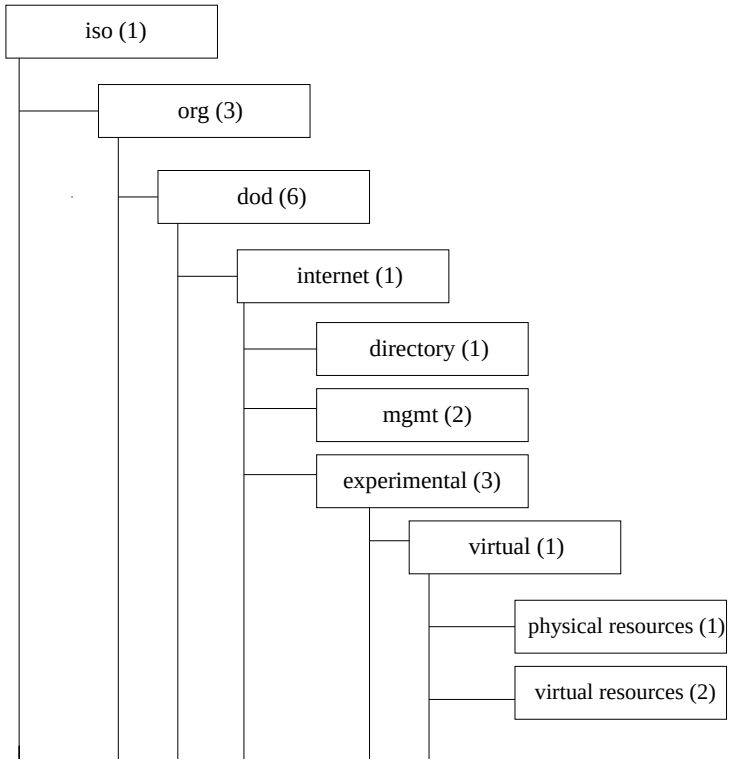

Figura 30 – Definição do Objeto vMIB (virtual).

A vMIB contrariamente às MIB´s para Xen e VMware ESX apresentadas, possui dois grupos distintos: *physicalResources* e *virtualResources*. A VMWARE-ROOT-MIB, por exemplo, possui distinção entre objetos reais e virtuais implementada através de tabelas dispostas nos grupos, onde os objetos que estão fora da tabela são físicos e os que compõem a mesma virtuais. Tal afirmativa pode ser analisada na Figura 26 e na Tabela 16, por exemplo.

A vMIB procura distinguir os objetos em físicos e virtuais visando, principalmente, a não

replicar objetos com mesma funcionalidade, como por exemplo objetos que descrevem a identificação da máquina virtual. Esse tipo de objeto também está presente na MIB para VMware ESX, tal afirmativa pode ser observada nos quatro grupos que compõe a estrutura da VMWARE-ROOT-MIB, sendo que os objetos são *CpuVmID*, *memVmID*, *hbaVmId* e *netVmId*.

Como falamos anteriormente, a vMIB é distribuída inicialmente em dois grupos, *physicalResources* e *virtualResources*. Para melhor observarmos, os grupos estão descritos conforme o último nível da Figura 30 e formam respectivamente as estruturas 1.3.6.1.3.1.1 e 1.3.6.1.3.1.2. Todos os objetos na vMIB serão hierarquicamente descendentes desses dois grupos e possuem esta estrutura de identificação primária.

4.2 Grupo *physicalResources*

A estrutura *physicalResources* procura descrever os recursos físicos que compõem a estrutura que suporta as máquinas virtuais. A árvore hierárquica com os objetos principais deste grupo é apresentada na Figura 31, onde podemos observar os objetos descritos em quatro grupos *processor*, *memory*, *disk* e *network*. Tais objetos dividem a estrutura conforme os recursos de processamento, memória, disco e rede, respectivamente.

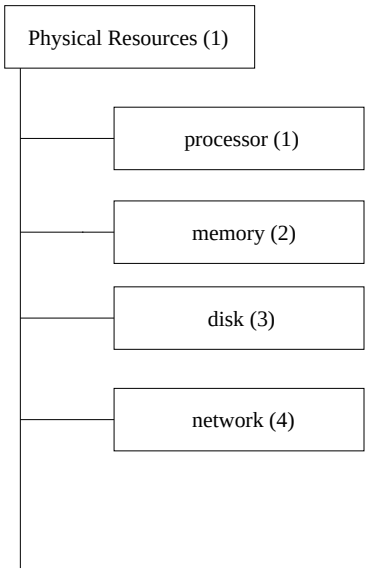

Figura 31 – Grupo *physical Resources*.

O grupo *processor* é composto por objetos que definem as características gerais da CPU do sistema que hospeda as máquinas virtuais. Especificamente através dele podemos obter informações a respeito da quantidade e tipo de CPU´s , por exemplo. O grupo *processor* pode ser analisado no formato de árvore através da Figura 32 e a descrição dos objetos que compõem este grupo pode ser observada na Tabela 19.

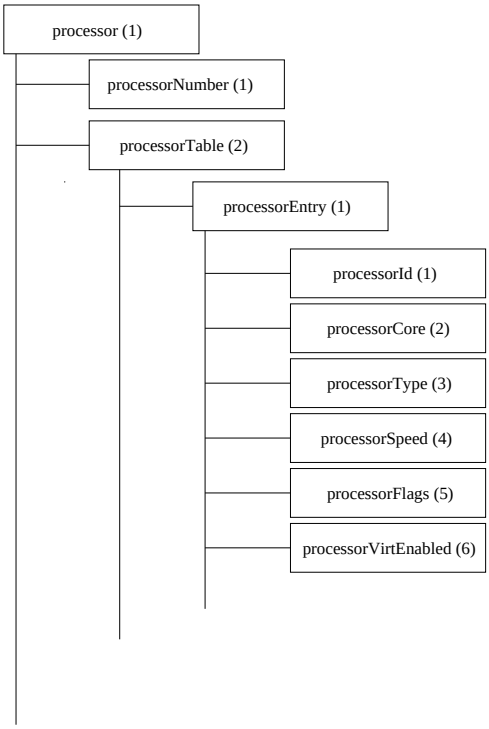

Figura 32 – Grupo *processor*.

Tabela 19 – Tabela de descrição dos objetos do grupo *processor*.

Objeto	Sintaxe	Acesso	Descrição
processorNumber	Integer	RO	Quantidade de CPU´s.
processorTable	Sequence	NA	Tabela do grupo CPU.
processorEntry	processorEntry	NA	Entrada da tabela.
processorid	Integer	RO	Identificação do processador.
processorCore	Integer	RO	Quantidade de núcleos do processador.
processorType	DisplayString	RO	Descrição do tipo do processador.
processorSpeed	Integer	RO	Capacidade do processador(Mhz).
processorFlags	DisplayString	RO	*Flags* de controle .
processorVirtEnabled	Integer	RO	Suporte do processador a virtualização.

O grupo *memory* é composto por objetos que definem as características do recurso de memória do sistema que hospeda as máquinas virtuais. Especificamente através dele podemos obter informações a respeito da quantidade de memória disponível, por exemplo. O grupo *memory* pode ser analisado no formato de árvore através da Figura 33, e a descrição dos objetos que compõe este grupo pode ser observada na Tabela 20.

Tabela 20 – Tabela de descrição dos objetos do grupo *memory*.

Objeto	Sintaxe	Acesso	Descrição
memoryTotal	Integer	RO	Quantidade de memória.
memoryFree	Integer	RO	Quantidade de memória disponível.

O grupo *disk* é composto por objetos que definem as características do recurso de disco

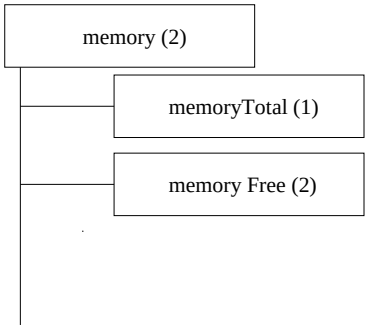

Figura 33 – Grupo *memory*.

do sistema que hospeda as máquinas virtuais. Especificamente através dele podemos obter informações a respeito do nome do disco, por exemplo. O grupo *disk* pode ser analisado no formato de árvore através da Figura 34, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 21.

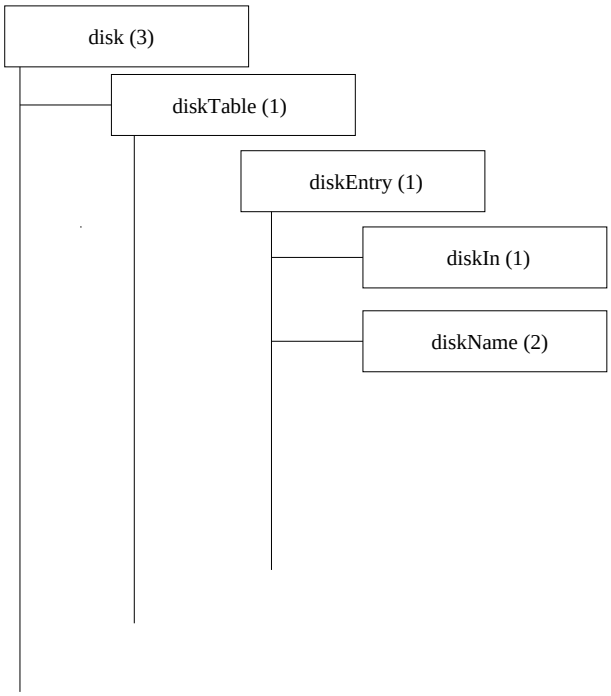

Figura 34 – Grupo *disk*.

Tabela 21 – Tabela de descrição dos objetos do grupo *disk*.

Objeto	Sintaxe	Acesso	Descrição
diskTable	Sequence	NA	Tabela do grupo *disk*.
diskEntry	diskEntry	NA	Entrada da tabela.
diskIn	Integer	RO	Index da tabela.
diskName	DisplayString	RO	Nome do disco.

O grupo *network* é composto por objetos que definem as características dos recursos de rede do sistema que hospeda as máquinas virtuais. Especificamente através dele podemos obter

informações a respeito da quantidade de kbytes que são enviados e recebidos, por exemplo. O grupo *network* pode ser analisado no formato de árvore através da Figura 35, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 22.

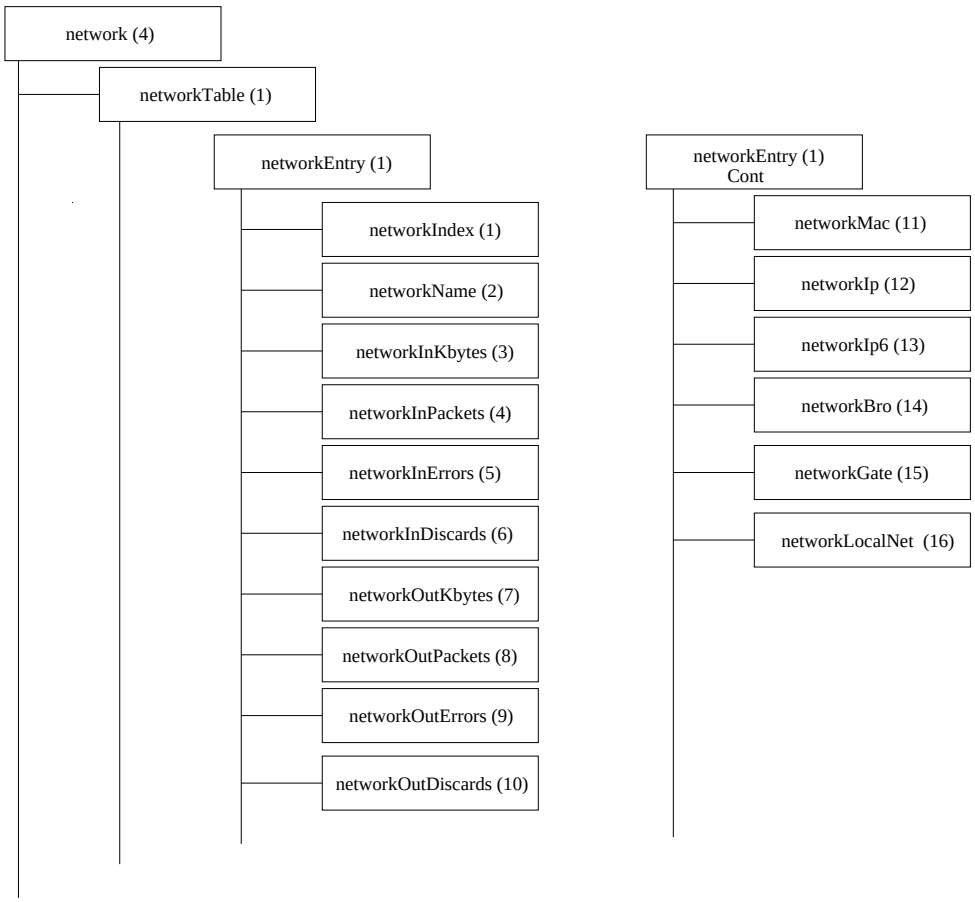

Figura 35 – Grupo *network*.

4.3 Grupo *virtualResources*

A estrutura *virtualResources* procura descrever os recursos virtuais disponíveis nas máquinas virtuais. A árvore hierárquica com os objetos principais deste grupo é apresentada na Figura 36, onde podemos observar os objetos descritos em dois grupos *virtualMachine*, *virtualMachineMonitor*; o primeiro apresenta informações sobre as máquinas virtuais e o segundo descreve informações gerais sobre o sistema que implementa a virtualização. Para melhor explicarmos descreveremos os objetos que compõem estes grupos a seguir.

O grupo *virtualMachine* é composto por uma tabela que contém informações sobre as máquinas virtuais e por mais dois objetos *virtualMachineName* e *virtualMachineStatus*.

A estrutura utilizando uma tabela foi escolhida devido a flexibilidade que a tabela proporciona na forma de acesso aos objetos da MIB, no caso da vMIB o principal objeto (*virtualV-*

Tabela 22 – Tabela de descrição dos objetos do grupo *network*.

Objeto	Sintaxe	Acesso	Descrição
networkTable	Sequence	NA	Tabela do grupo *network*.
networkEntry	networkEntry	NA	Entrada da tabela.
networkIndex	Integer	RO	Index da tabela.
networkName	DisplayString	RO	Nome da inteface de rede.
networkInKbytes	Counter	RO	Kbytes recebidos pela interface.
networkInPackets	Counter	RO	Pacotes recebidos pela interface.
networkInErrors	Counter	RO	Pacotes com erro recebidos pela interface.
networkInDiscards	Counter	RO	Pacotes descartados pela interface.
networkOutKbytes	Counter	RO	Kbytes enviados pela interface.
networkOutPackets	Counter	RO	Pacotes enviados pela interface.
networkOutErrors	Counter	RO	Pacotes não enviados devido a erros.
networkOutDiscards	Counter	RO	Pacotes que seriam enviados,descartados.
networkMac	DisplayString	RO	Endereço MAC da interface.
networkIp4	ipaddress	RO	Endereço IPv4.
networkIp6	DisplayString	RO	Endereço IPv6.
networkBro	ipaddress	RO	Endereço IP para *Broadcast*.
networkGate	DisplayString	RO	Nome do gateway da rede.
networkLocalNet	ipaddress	RO	Endereço IP da rede local.

mIdVm) desta tabela descreve um nível acima dos demais objetos a qual máquina virtual está sendo solicitada a operação de leitura ou escrita, definindo assim a máquina virtual a que se destina a operação. Esse tipo de implementação elimina a necessidade de identificação replicada do objeto ID da máquina virtual, característica esta encontrada nas MIB´s utilizadas nos sistemas Xen e VMware apresentadas anteriomente.

Os objetos *virtualMachineName* e *virtualMachineStatus*, definem respectivamente o nome da máquina virtual e o status da mesma que pode ser por exemplo *running* (1), *blocked* (2), *paused* (3) ou *shutdown* (4).

O objeto *virtualVmIdVm* forma um grupo composto pelos objetos que fornecem informações provenientes dos recursos virtuais disponíveis nas máquinas virtuais. A árvore hierárquica composta pelos objetos que fazem parte do grupo *virtualResources* e seguem o ramo *virtual Machine* é apresentada na Figura 37, onde podemos analisar também os recursos descritos em quatro grupos *virtualProcessor*, *virtualMemory*, *virtualDisk* e *virtualNetwork*, tais objetos dividem a estrutura conforme os recursos de processamento virtual, memória virtual, disco virtual e rede virtual respectivamente.

O grupo *virtualProcessor* é composto por objetos que definem as características gerais das CPU´s virtuais que estão em funcionamento na máquinas virtuais. Especificamente através dele podemos obter informações a respeito da quantidade, identificação e alterar a quantidade de CPU´s, por exemplo. O grupo *virtualProcessor* pode ser analisado no formato de árvore através da Figura 38, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 23.

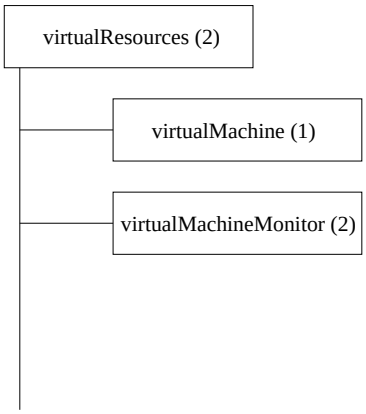

Figura 36 – Grupo *virtual Resources*.

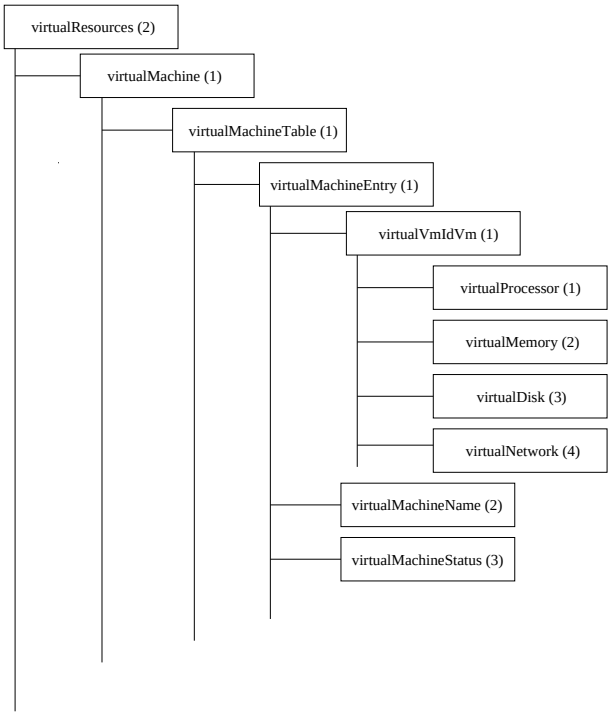

Figura 37 – Grupo *virtual Machine*.

Tabela 23 – Tabela de descrição dos objetos do grupo *virtual Processor*.

Objeto	Sintaxe	Acesso	Descrição
virtualProcessorNumber	Integer	RW	Processadores virtuais.
virtualProcessorTable	Sequence	NA	Tabela do grupo.
virtualProcessorEntry	virtualProcessorEntry	NA	Entrada da tabela.
virtualProcessorId	Integer	RO	ID do processador virtual.

O grupo *virtualMemory* é composto por objetos que disponibilizam informações a respeito do recurso de memória alocado nas memória virtuais. Especificamente através dele podemos obter informações a respeito da quantidade de memória utilizada e disponível em cada máquina virtual gerenciada, sendo que também podemos realizar a alteração das configurações de me-

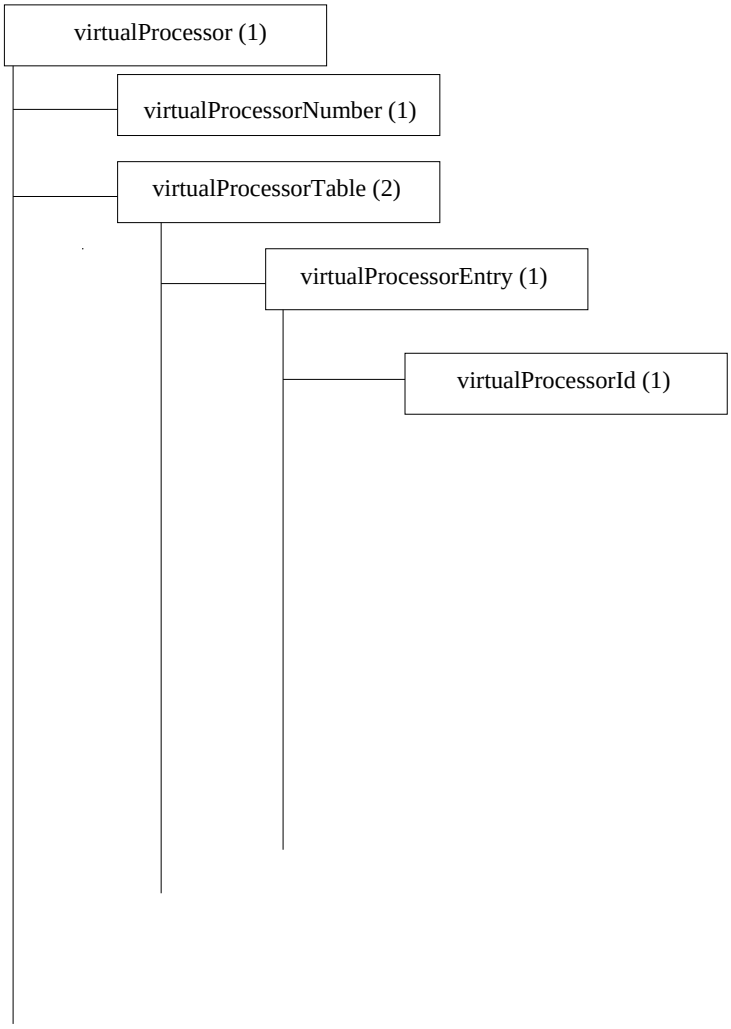

Figura 38 – Grupo *virtual Processor*.

mória. O grupo *virtualMemory* pode ser analisado no formato de árvore, através da Figura 39, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 24.

Tabela 24 – Tabela de descrição dos objetos do grupo *virtual Memory*.

Objeto	Sintaxe	Acesso	Descrição
virtualMemoryTable	Sequence	NA	Tabela do grupo.
virtualMemoryEntry	virtualMemoryEntry	NA	Entrada da tabela.
virtualMemoryTotal	Integer	RW	Memória disponível.
virtualMemoryUsed	Integer	RW	Memória Utilizada.

O grupo *virtualDisk* é composto por objetos que disponibilizam informações a respeito do recurso de disco alocado nas memória virtuais. Especificamente através dele podemos obter informações a respeito da quantidade de disco disponível em cada máquina virtual gerenciada, por exemplo. O grupo *virtualDisk* pode ser analisado no formato de árvore, através da Figura 40, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 25.

O grupo *virtualNetwork* é composto por objetos que definem as características dos recursos

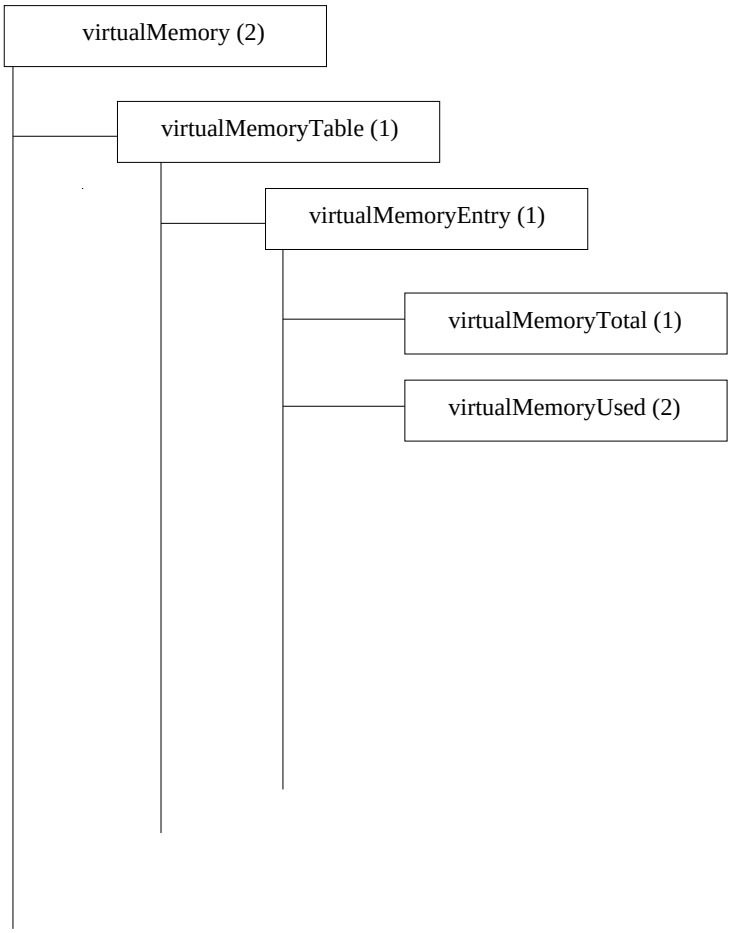

Figura 39 – Grupo *virtual Memory*.

Tabela 25 – Tabela de descrição dos objetos do grupo.

Objeto	Sintaxe	Acesso	Descrição
virtualDiskTable	Sequence	NA	Tabela do grupo *virtualDisk*.
virtualDiskEntry	virtualDiskEntry	NA	Entrada da tabela.
virtualDiskId	Integer	RO	Identificação do disco.
virtualDiskName	DisplayString	RO	Nome do disco.

de rede disponíveis nas máquinas virtuais. Especificamente através dele podemos obter informações a respeito da quantidade de kbytes que são enviados e recebidos pelas interfaces de rede virtuais, por exemplo. O grupo *virtualNetwork* pode ser analisado no formato de árvore, através da Figura 41, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 26.

O grupo *virtualMachineMonitor* é composto por objetos que definem as características gerais do sistema virtual que está gerenciando as máquinas virtuais. Especificamente através dele podemos obter informações a respeito da versão do sistema virtual, por exemplo. O grupo *virtualMachineMonitor* pode ser analisado no formato de árvore, através da Figura 42, e a descrição dos objetos que compõe esse grupo pode ser observada na Tabela 27.

A vMIB como podemos notar possui estrutura baseada nos padrões SMI e baseia-se em

Tabela 26 – Tabela de descrição dos objetos do grupo *virtual Network*.

Object	Syntax	Access	Description
virtualNetworkTable	Sequence	NA	Tabela do grupo.
virtualNetworkEntry	virtualNetworkEntry	NA	Entrada da tabela.
virtualNetworkIndex	Integer	RO	Index da tabela.
virtualNetworkName	DisplayString	RO	Nome da interface de rede.
virtualNetworkInKbytes	Counter	RO	Kbytes recebidos.
virtualNetworkInPackets	Counter	RO	Pacotes recebidos.
virtualNetworkInErrors	Counter	RO	Pacotes com erro recebidos.
virtualNetworkInDiscards	Counter	RO	Pacotes descartados.
virtualNetworkOutKbytes	Counter	RO	Kbytes enviados.
virtualNetworkOutPackets	Counter	RO	Pacotes enviados.
virtualNetworkOutErrors	Counter	RO	Pacotes não enviados, com erros.
virtualNetworkOutDiscards	Counter	RO	Pacotes enviados,descartados .
virtualNetworkMac	DisplayString	RO	Endereço MAC da interface.
virtualNetworkIp4	ipaddress	RO	Endereço IPv4.
virtualNetworkIp6	DisplayString	RO	Endereço IPv6.
virtualNetworkBro	ipaddress	RO	Endereço IP para *Broadcast*.
virtualNetworkGate	DisplayString	RO	Nome do gateway da rede.
virtualNetworkLocalNet	ipaddress	RO	Endereço IP da rede local.

Tabela 27 – Tabela de descrição dos objetos do grupo *virtual Machine Monitor*.

Object	Syntax	Access	Description
virtualMachineMonitorSystem	DisplayString	RO	Sistema Virtual.
virtualMachineMonitorVersion	Integer	RO	Versão do Sistema.

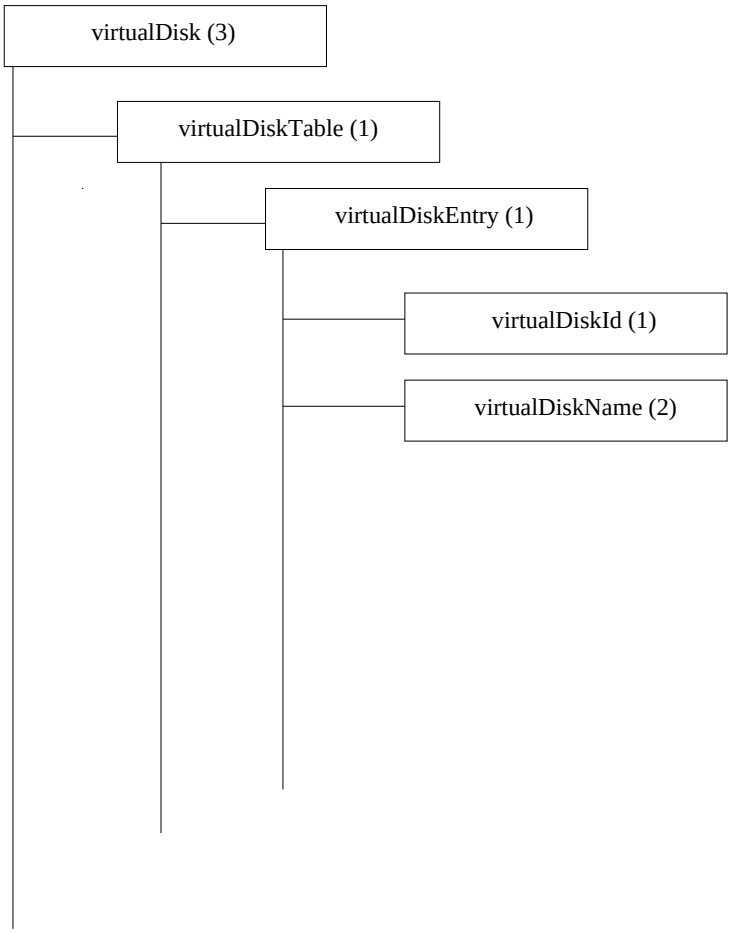

Figura 40 – Grupo *virtual Disk*.

grande parte na MIB-II, e nas MIB´s desenvolvidas para os sistemas Xen e VMware que apresentamos. Porém, acrescenta a capacidade de gerência de máquinas virtuais e possui uma estrutura genérica e flexível permitindo que a mesma base de informações possa ser utilizada para gerenciar qualquer sistema virtual.

Ressaltamos também que os objetos genéricos da vMIB podem ser encontrados em praticamente todos os sistemas virtuais, e devido a este fator proporciona padronização, flexibilidade e principalmente a capacidade de gerenciar, em um ambiente heterogêneo, sistemas virtuais diferentes.

Porém a generalização termina por restringir a quantidade de objetos na vMIB, fator este que termina influenciando também na quantidade de objetos que podem ser controlados utilizando a vMIB. Outro fator que contribui para a definição destes poucos objetos de controle é o fato do ambiente virtual ser um ambiente limitado e, portanto, possui em sua maioria, uma quantidade reduzida de recursos.

A vMIB torna-se portanto uma ferramenta importante quando utilizada para gerência de máquinas virtuais, porém necessita ser utilizada em um ambiente que proporcione os mais variados testes possíveis, com o intuito de validar sua eficaz utilização.

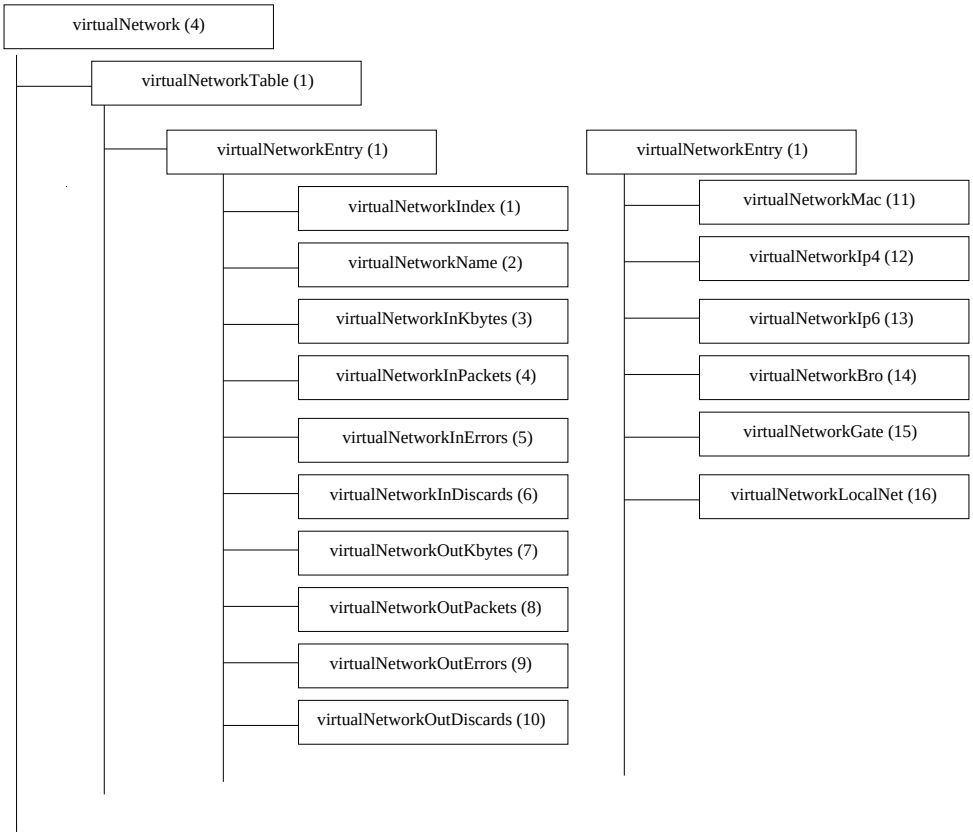

Figura 41 – Grupo *virtual Network*.

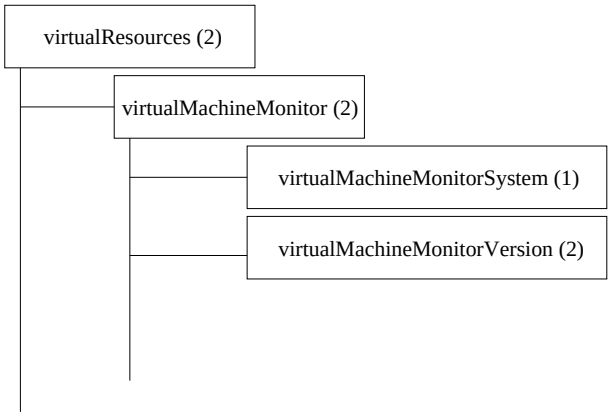

Figura 42 – Grupo *virtual Machine Monitor*.

5 Validação

Com o intuito de validar a base de informações desenvolvida foram criados dois agentes, o primeiro foi desenvolvido para operar sobre a plataforma Xen e o segundo foi desenvolvido para trabalhar com o sistema OpenVZ.

Visando a padronizar o desenvolvimento de ambos foram utilizadas ferramentas de gerência desenvolvidas para cada um dos sistemas virtuais, no caso do Xen utilizou-se a ferramenta XM, e no caso do OpenVZ utilizou-se a ferramenta VZCTL, ambas de distribuição livre. Essas ferramentas atualmente são as mais utilizadas para gerência destes sistemas virtuais e possuem a capacidade de monitorar e controlar os recursos dos mesmos. Os agentes foram desenvolvidos utilizando a linguagem Shell Script reproduzindo comandos disponíveis para gerência disponíveis em cada ferramenta.

O Xen possui uma ferramenta de gerência chamada XM (*Xen Management*) que realiza operações em seu ambiente, como por exemplo a alteração de quantidade de memória alocada e utilizadas nas máquinas virtuais, a capacidade listar os domínios e seus respectivos recursos disponíveis, alterar a quantidade de CPU´s virtuais, criar e destruir interfaces de rede, seja ele dom0 ou domU [45].

A estrutura básica de um comando XM é descrita abaixo:

- xm <*subcommand*> <*domain-id*> [*OPTIONS*]

O agente desenvolvido para o sistema Xen utiliza a ferramenta XM para gerência dos recursos dos domínios. Abaixo, no Exemplo 1 e Exemplo 2, podemos observar a descrição do algoritmo que expressa a seleção do tipo de operação realizada, *Get* para leitura e *Set* para escrita. Esse bloco define conforme a seleção e a identificação do objeto o comando XM que irá realizar a operação, seja esta de leitura ou de escrita. Abaixo, no Exemplo 1, apresentamos primeiramente um algoritmo genérico de uma operação de leitura em uma máquina virtual.

```
Exemplo 1.

1.se operação ‘‘get’’
2.   caso ‘‘Objeto’’
3.        executar linha de comando XM
4.        retornar valor do objeto para o gerente
5.        retornar o tipo do objeto para o gerente
```

```
6.        retornar o valor resultante da linha de comando
7.    fimcaso
8.fimse
```

No Exemplo 1, apresentamos uma operação de leitura utilizando o agente Xen. Destacamos na linha 1 a opção "*get*", que é responsável por definir para a estrutura do algoritmo uma operação de leitura. Na linha 2, notamos a definição do objeto, que pode ser por exemplo 1.3.6.1.3.1 (vMIB), responsável por receber a informação que define o objeto que é alvo da operação. Na linha 3, notamos a linha de comando da ferramenta XM responsável por executar a operação no objeto selecionado na linha 2, na continuação do bloco de algoritmo, relativas as linhas 4,5 e 6 notamos a definição dos valores que vão ser o retorno da informação para o gerente representada pela confirmação da identificação e tipo de objeto e o valor resultante da linha 3. O conteúdo do Exemplo 1 é genérico e, portanto, replicado para cada objeto que recebe uma operação de leitura no agente Xen.

```
Exemplo 2.
```

```
1.se operação ‘‘set’’
2.   caso ‘‘Objeto’’
3.        executar linha de comando XM
4.        retornar valor do objeto para o gerente
5.        retornar o tipo do objeto para o gerente
6.        retornar o valor resultante da linha de comando
7.    fimcaso
8.fimse
```

No Exemplo 2, apresentamos uma operação de escrita utilizando o agente Xen. Destacamos na linha 1 a opção "*set*", que é responsável por definir para a estrutura do algoritmo uma operação de escrita. Na linha 2, notamos a definição do objeto, responsável por receber a informação que define o objeto que é alvo da operação. Na linha 3, notamos a linha de comando da ferramenta XM responsável por executar a operação no objeto selecionado na linha 2, na continuação do bloco de algoritmo, relativas as linhas 4,5 e 6 notamos a definição dos valores que vão ser o retorno da informação para o gerente representada pela confirmação da identificação e tipo de objeto e o valor resultante da linha 3. O conteúdo do Exemplo 2 é genérico e, portanto, replicado para cada objeto que recebe uma operação de escrita no agente Xen.

Visando testar o agente xen foi utilizada uma estrutura de hardware com as seguintes características:

- Processador: Pentium 4 2.8GHz 1MB de cache;
- Memória: 2.5GB de memória RAM - 2X 1GB DDR PC2700 333Mhz + 2X 256MB DDR PC 2700 333MHz;
- Disco : 40 GB IDE;
- Placa de Rede: Onboard Gigabit.

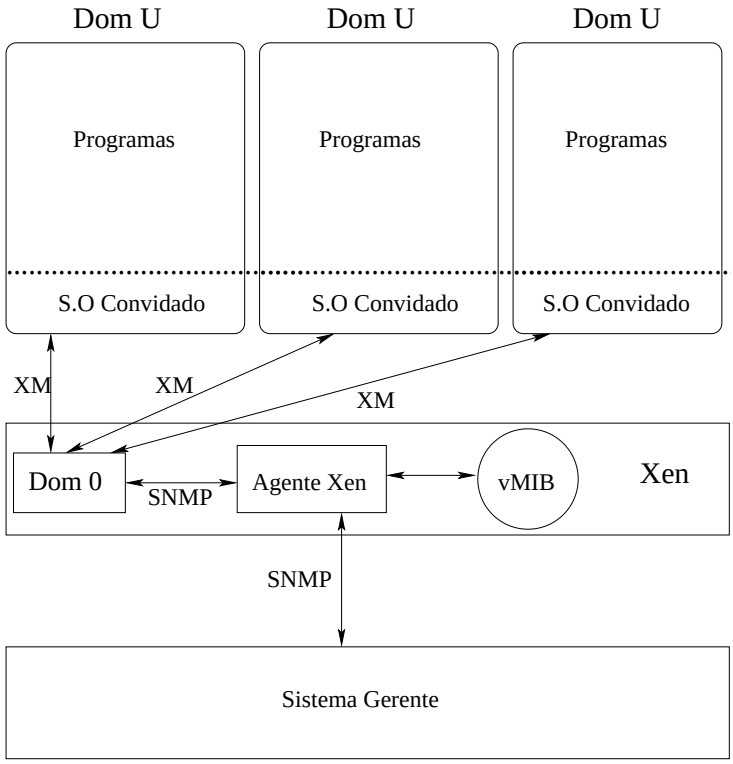

Figura 43 – Cenário de teste do Agente Xen.

Essa estrutura de hardware esta instalada sobre o sistema operacional Debian 4.0, utilizando o Xen 3.4 e a ferramenta XM 2.0. Sobre esta estrutura de hardware e software foram criadas três máquinas virtuais. A Figura 43 ilustra o ambiente de teste criado para o agente Xen, utilizando a ferramenta XM para gerenciamento das máquinas virtuais criadas. Nesta Figura destacamos o sistema gerente realizando solicitações através do protocolo SNMP para o agente Xen, que utilizando-se das informações da vMIB acessa o Dom 0 e através da ferramanta XM gerencia as máquinas virtuais alocadas no sistema. Utilizando desta metodologia todos os objetos presentes na vMIB foram testados pelo menos dez vezes através de chamadas SNMP realizadas pelo sistema gerente e funcionaram corretamente.

Cabe ressaltar também que o agente desenvolvido para o sistema Xen também foi utilizado na dissertação com tema “Uma Arquitetura de Gerência de Rede de Máquinas Virtuais com ênfase na Emulação” desenvolvida pelo discente Mauro Storch do Programa de Pós-Graduação em Ciência da Computação – PPGCC/PUCRS, sob orientação do Profº César A. F. De Rose [46].

No trabalho, em questão, o agente foi utilizado como ferramenta de apoio na realização de configuração de máquinas virtuais com o intuito de controlar parâmetros de conexões de rede entre as mesmas. Para tanto todos os objetos presentes nas estruturas de rede, tanto real como virtual foram exaustivamente utilizados para gerência destes recursos e funcionaram corretamente.

O OpenVZ possui uma ferramenta de gerência chamada VZCTL que realiza operações em seu ambiente, como por exemplo, a alteração de quantidade de memória alocada e utilizadas nas máquinas virtuais, a capacidade listar os domínios e seus respectivos recursos disponíveis, alterar a quantidade de CPU´s virtuais, criar e destruir interfaces de rede [47].

A estrutura básica de um comando VZCTL é descrita abaixo:

- VZCTL <*subcommand*> <*VE-id*> [*PARAMETERS*]

O agente desenvolvido para o sistema OpenVZ utiliza a ferramenta VZCTL para gerência dos recursos dos domínios. Abaixo, no Exemplo 3, podemos observar a descrição de um bloco do algoritmo que expressa a seleção do tipo de operação realizada, *Get* para leitura e *Set* para escrita. Este bloco define conforme a seleção e a identificação do objeto o comando VZCTL que irá realizar a operação, seja esta de leitura ou de escrita. No Exemplo 3, apresentamos um modelo de uma operação de escrita de genérica utilizando um agente desenvolvido para o sistema OpenVZ.

```
Exemplo 3.

1.se operação ‘‘set’’
2.   caso ‘‘Objeto’’
3.        executar linha de comando VZCTL
4.        retornar valor do objeto para o gerente
5.        retornar o tipo do objeto para o gerente
6.        retornar o valor resultante da linha de comando
7.    fimcaso
8.fimse
```

No Exemplo 3, utilizando o agente OpenVZ destacamos na linha 1 a opção “*set*”, que é responsável por definir para a estrutura do programa uma operação de escrita. Na linha 2, notamos a definição do objeto, responsável por receber a informação que define o objeto que é alvo da operação. Na continuação do bloco do algoritmo representada pelas linhas 4,5 e 6 observamos a definição do que vai ser o retorno da informação para o gerente, representada respectivamente pela confirmação da identificação e tipo de objeto e pelo valor resultante da linha de comando do VZCTL. O conteúdo do Exemplo 3 é genérico e, portanto, replicado para cada objeto que recebe uma operação de escrita no agente para OpenVZ.

A estrutura de hardware utilizada para testes no agente OpenVZ é a mesma utilizada para o agente Xen e esta instalada sobre o sistema operacional Debian 4.0, utilizando o OpenVZ 2.6.8 e a ferramenta VZCTL 3.0. Sobre esta estrutura de hardware e software foram criadas três máquinas virtuais. A Figura 44 ilustra o ambiente de teste criado para o agente OpenVZ, utilizando a ferramenta VZCTL para gerenciamento das máquinas virtuais criadas. Nesta Figura destacamos o sistema gerente realizando solicitações através do protocolo SNMP para o agente OpenVZ, que utilizando-se das informações da vMIB acessa a ferramanta VZCTL e gerencia as máquinas virtuais alocadas no sistema. Utilizando desta metodologia os objetos presentes nos grupos *processor, memory, virtualprocessor* e *virtualmemory* da vMIB foram testados pelo menos dez vezes através de chamadas SNMP realizadas pelo sistema gerente e funcionaram corretamente.

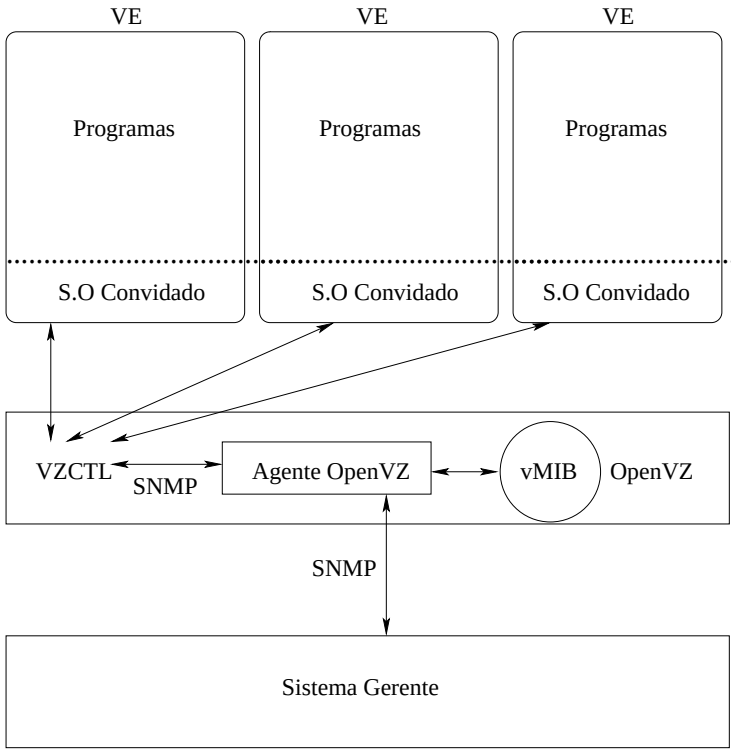

Figura 44 – Cenário de teste do Agente OpenVZ.

As ferramentas XM e VZCTL foram escolhidas pelo fato de possuirem sintaxe e funcionalidades semelhantes o que facilitou a implementação dos agentes, visto que como observamos pode-se manter uma padronização na construção dos agentes, padronização esta que pode ser seguida no caso de implemetação de um novo agente para qualquer outro sistema virtual, visto que de modo geral as ferramentas de gerência para este tipo de sistema são semelhantes.

Atenção: caso seja necessário o desenvolvimento de um agente para o sistema VMware ESX recomenda-se a utilização da ferramenta VMware-Console, visto que esta também possui funcionalidades semelhantes ao XM e OpenVZ [28].

Utilizando o VMware-Console poderíamos possuir o seguinte cenário de implementação do agente, representado pelo Exemplo 4.

```
Exemplo 4.

1.se operação ‘‘set’’
2.   caso ‘‘Objeto’’
3.        executar linha de comando VMWARE-CONSOLE
4.        retornar valor do objeto para o gerente
5.        retornar o tipo do objeto para o gerente
6.        retornar o valor resultante da linha de comando
7.    fimcaso
8.fimse
```

Outro fator que pesou na escolha das ferramentas foi a finalidade de utilização dos mesmos, a validação da MIB, sendo que a questão pertinente ao desempenho não é considerada para a validação. Caso necessária implementação visando ao desempenho recomenda-se o desenvolvimento de agentes implemetados diretamente sobre bibliotecas, como por exemplo a XenApi do sistema xen [48]. Cabe ressaltar também que a ferramenta XM foi implementada utilizando a XenAPI, porém possui um nível de abstração superior.

Analisando os Exemplos 1,2,3 e 4 notamos que a implementação dos agentes é padronizada e simples, tornando-se fácil de adaptar a qualquer tipo de sistema virtual que necessite ser gerenciado utilizando o protocolo SNMP.

6 Conclusão

Como observamos neste trabalho, máquinas virtuais apresentam-se atualmente como soluções para os mais diversos tipos de aplicações, principalmente por sua capacidade de prover isolamento, migração e consolidação. Porém visando obter uma melhor utilização destes recursos necessita-se de ferramentas que realizem a tarefa de gerência de máquinas virtuais.

Observamos também que utilizando o protocolo SNMP atualmente possuímos ferramentas limitadas para gerência desse tipo de ambiente, o que torna a atividade de gerência restrita a uma pequena quantidade de recursos, que na sua maioria permitem apenas a atividade de monitoração, como é o caso das bases de informações desenvolvidas para os sistemas virtuais Xen e Vwware apresentado neste trabalho.

Este trabalho apresenta o desenvolvimento e a utilização de uma MIB genérica, elaborada para o gerenciamento de máquinas virtuais, o que acarreta em maior eficiência e flexibilidade no gerenciamento de recursos virtuais.

A vMIB acrescenta maior flexibilidade na gerência de recursos virtuais, pois utilizada com agentes específicos possui a capacidade de interoperar a tarefa de gerência entre sistemas virtuais diferentes. Esta característica possibilita também que esta MIB genérica seja utilizada simultaneamente para gerenciar um ambiente heterogêneo composto por diversas máquinas virtuais, isoladas, trabalhando com sistemas virtuais diferentes, e provendo suporte aos mais variados tipos de aplicações.

A Eficiência da vMIB está ligada a capacidade de distinção de recursos pertences ao monitor de máquinas virtuais (VMM) e aos recursos alocados sobre as máquinas virtuais, descritos como *physical Resources* e *virtual Resources* respectivamente. Porém, se tratando de uma ferramenta genérica não contempla todos os recursos disponíveis nos mais diversos tipos de sistemas virtuais.

A vMIB acrescenta a tarefa de gerenciamento de máquinas virtuais a capacidade de gerência de recursos virtuais independentemente do sistema virtual utilizado, pois como notamos MIB´s anteriores não possibilitavam a implementação da atividade de escrita nos agentes e são específicas para o gerenciamento deste ou daquele sistema virtual em particular. Com relação a esta característica cabe ressaltar que a quantidade limitada de recursos que podem ser controlados deve-se principalmente a generalização da MIB desenvolvida e ao fato do ambiente virtual possuir uma quantidade restrita de recursos.

A vMIB foi validada utilizando agentes desenvolvidos para os sistemas Xen e OpenVZ, utilizando ferramentas de gerência largamente testadas por estes sistema, destacamos também que a vMIB foi utilizada em outro projeto de pesquisa, citado no trabalho, provendo o gerenci-

amento de recursos de rede nos níveis real e virtual.

Concluimos, então, que a vMIB é uma ferramenta eficiente que pode ser facilmente utilizada para gerenciamento de máquinas virtuais, independentemente, do sistema virtual instalado.

Referências

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

[2] TANENBAUM, A. S.; WOODHULL, A. S. *Sistemas Operacionais Projeto e Implementação*. 2^a. Porto Alegre: Bookman, 2000.

[3] NEIGER, G. et al. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, Intel, Santa Clara, CA, USA, v. 10, n. 3, p. 167–177, aug. 2006.

[4] MENASCÉ, D. A. Virtualization: Concepts, applications, and performance modeling. In: *Computer Measurement Group Conference, 31, 2005*. Orlando: **Proceedings...** Turnersville, Computer Measurement Group, 2005. p. 142–149.

[5] APPARAO, P.; MAKINENI, S.; NEWELL, D. Characterization of network processing overheads in Xen. In: *International Workshop on Virtualization Technology in Distributed Computing, 2, 2006*. Washington: **Proceedings...** New York, IEEE Computer Society, 2006. p. 2.

[6] ROSE, M. T. *An Introduction to Networking Management*. 2nd. ed. New York: Prentice Hall, Inc, 1996.

[7] SILBERSHATZ, A. *Sistemas Operacionais, Conceitos*. 5th. ed. São Paulo: Prentice Hall, 2000.

[8] LAUREANO, M. A. P. et al. Detecção de intrusão em máquinas virtuais. In: *Simpósio de Segurança em Informática, 5, 2003*. São José dos Campos: **Anais...** Campinas, UNICAMP, 2003. p. 101–107.

[9] AMD. *AMD64 Virtualization Codenamed "Pacifica" Technology Secure Virtual Machine Architecture Reference Manual*. 2006. Disponível em: <`http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/33047.pdf`>. Acesso em: 14 out. 2007.

[10] CRAIG, I. D. *Virtual Machines*. 1st. ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 1852339691.

[11] The VMWARE Team. *THE VMWARE Project*. 2007. Disponível em: <`http://www.vmware.com`>. Acesso em: 23 set. 2007.

[12] KIDDLE, C.; SIMMONDS, R.; UNGER, B. Improving scalability of network emulation through parallelism and abstraction. In: *Annual Symposium on Simulation, 38, 2005*. Washington: **Proceedings...** New York, IEEE Computer Society, 2005. p. 119–129.

[13] The OpenVZ Team. *THE OpenVZ Project*. 2007. Disponível em: <http://www.openvz.org>. Acesso em: 25 nov. 2007.

[14] T.PERKINS, D. *RMON: Remote Monitoring of SNMP-Managed LANs*. 1st. ed. Upper Saddle River, NJ: Prentice Hall, Inc, 1999.

[15] LABORDE, R. et al. Implementation of a formal security policy refinement process in WBEM architecture. *J. Netw. Syst. Manage.*, Plenum Press, New York, NY, USA, v. 15, n. 2, p. 241–266, feb. 2007. ISSN 1064-7570.

[16] JOB, J. C. C.; SIMõES, M. A. C. *Gerenciamento de Sistemas Baseado no Padrão WBEM*. 2002. Disponível em: <http://www.sbc.org.br/reic/edicoes/2002e4/>. Acesso em: 12 ago. 2007.

[17] GUIAGOUSSOU, M.; BOUTABA, R.; KADOCH, M. A java API for advanced faults management. In: *Integrated Network Management Proceedings, 1, 2001*. Seattle: **Proceedings...** New York, IEEE Computer Society, 2001. p. 483–498.

[18] Distributed Management Task Force. *WBEM,Web-Based Enterprise Management*. 2007. Disponível em: <http://www.dmtf.org>. Acesso em: 6 out. 2007.

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

[20] CHERKASOVA, L.; GARDNER, R. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In: *Annual Technical Conference, 2, 2005*. Fort Collins: **Proceedings...** New York, ACM Press, 2005. p. 387–390.

[21] MATTHEWS, J. N. et al. Quantifying the performance isolation properties of virtualization systems. In: *Workshop on Experimental Computer Science, 3, 2007*. New York: **Proceedings...** New York, ACM Press, 2007. p. 6.

[22] BARHAM, P. et al. Xen and the art of virtualization. In: *Symposium on Operating Systems Principles, 19, 2003*. Bolton Landing: **Proceedings...** New York, ACM Press, 2003. p. 164–177.

[23] MENON, A. et al. Diagnosing performance overheads in the Xen virtual machine environment. In: *International Conference on Virtual execution environments, 1, 2005*. New York: **Proceedings...** New York, ACM Press, 2005. p. 13–23.

[24] YOUSEFF, L. et al. Evaluating the performance impact of Xen on MPI and process execution for HPC systems. In: *International Workshop on Virtualization Technology in Distributed Computing, 2, 2006*. Washington: **Proceedings...** New York, IEEE Computer Society, 2006. p. 1.

[25] GUPTA, D.; GARDNER, R.; CHERKASOVA, L. *XenMon: QoS Monitoring and Performance Profiling Tool*. HP (Hewlett-Packard) - Laboratories Palo Alto, CA, USA, Dez 2005.

[26] GUPTA, D. et al. Enforcing performance isolation across virtual machines in Xen. In: *International Middleware Conference, 7, 2006*. Melbourne: **Proceedings...** New York, Springer, 2006. p. 342–362.

[27] WALDSPURGER, C. A. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 36, n. SI, p. 181–194, 2002. ISSN 0163-5980.

[28] The VMWARE Team. *VMWARE Scripting API*. 2007. Disponível em: <http://www.vmware.com/support/developer/scripting-API/doc/Scripting_API.pdf>. Acesso em: 29 ago. 2007.

[29] MATTHEWS, J. N. et al. Quantifying the performance isolation properties of virtualization systems. In: *Workshop on Experimental Computer Science, 3, 2007*. New York: **Proceedings...** New York, ACM Press, 2007. p. 6.

[30] CHEN, Y.-C.; CHAN, I.-K. SNMP getrows: an effective scheme for retrieving management information from MIB tables. *Int. J. Netw. Manag.*, John Wiley & Sons, Inc., New York, NY, USA, v. 17, n. 1, p. 51–67, 2007. ISSN 1099-1190.

[31] Internet Engineering Task Force. *SMI RFC 1155: Structure and Identification of Management Information for TCP/IP- based internets*. 2007. Disponível em: <http://www.ietf.org/rfc/rfc1155.txt>. Acesso em: 12 out. 2007.

[32] Internet Engineering Task Force. *SMI RFC 1213: Structure and Identification of Management Information for TCP/IP- based internets*. 2007. Disponível em: <http://www.ietf.org/rfc/rfc1213.txt>. Acesso em: 12 out. 2007.

[33] Internet Engineering Task Force. *SMI RFC 1156: Structure and Identification of Management Information for TCP/IP- based internets*. 2007. Disponível em: <http://www.ietf.org/rfc/rfc1156.txt>. Acesso em: 12 out. 2007.

[34] MCCLOGHRIE, K.; KASTENHOLZ, F. *The Interfaces Group MIB*. 2000. Disponível em: <http://www.faqs.org/rfcs/rfc2863.html>. Acesso em: 17 out. 2007.

[35] MCCLOGHRIE, K.; KASTENHOLZ, F. *The Interfaces Group MIB using SMIv2*. 1997. Disponível em: <http://www.faqs.org/rfcs/rfc2233.html>. Acesso em: 17 out. 2007.

[36] ROSE, M. T. *Management Information Base for network management of TCP/IP-based internets: MIB-II*. 1990. Disponível em: <http://rfc.net/rfc1158.html>. Acesso em: 11 out. 2007.

[37] KIM, J.-K.; AHN, S.-J.; CHUNG, J.-W. Design and implementation of WAP-based LAN segment management system using RMON MIB. *Int. J. Netw. Manag.*, John Wiley & Sons, Inc., New York, NY, USA, v. 13, n. 3, p. 187–202, 2003. ISSN 1099-1190.

[38] KALBFLEISCH, C. et al. *RFC 2564-Application Management MIB*. 1999. Disponível em: <http://www.faqs.org/rfcs/rfc2564.html>. Acesso em: 11 out. 2007.

[39] Internet Engineering Task Force. *Host Resources MIB RFC 2790: Host Resources MIB*. 2000. Disponível em: <http://www.faqs.org/rfcs/rfc2790.html>. Acesso em: 19 out. 2007.

[40] The XEN-MIB Team. *TUBS-IBR-XEN-MIB DEFINITIONS*. 2007. Disponível em: <http://www.ibr.cs.tu-bs.de/svn/libsmi/trunk/mibs/tubs/TUBS-IBR-XEN-MIB>. Acesso em: 10 mar. 2007.

[41] The VMWARE-MIB Team. *VMWARE-ROOT-MIB DEFINITIONS*. 2007. Disponível em: <http://www.oidview.com/mibs/6876/VMWARE-ROOT-MIB.html>. Acesso em: 10 mar. 2007.

[42] TSAI, C.-H. et al. Virtualization-based techniques for enabling multi-tenant management tools. In: *International Workshop on Distributed Systems, 18, 2006*. San Jose: **Proceedings...** New York, IEEE Computer Society, 2007. p. 171–182.

[43] HUANG, W. et al. A case for high performance computing with virtual machines. In: *Annual International Conference on Supercomputing, 20, 2006*. Cairns: **Proceedings...** New York, ACM Press, 2006. p. 125–134.

[44] FIGUEIREDO, R. J.; DINDA, P. A.; FORTES, J. A. B. A case for grid computing on virtual machines. In: *International Conference on Distributed Computing Systems, 23, 2003*. Gainesville: **Proceedings...** New York, IEEE Computer Society, 2003. p. 550–559.

[45] The XenSource Team. *XM - Xen Management User Interface*. 2007. Disponível em: <http://linux.die.net/man/1/xm>. Acesso em: 21 out. 2007.

[46] STORCH, M.; DE ROSE, C. *Uma Arquitetura de Gerência de Rede de Máquinas Virtuais com ênfase na Emulação de Sistemas Distribuídos*. Centro de Pesquisa PUCRS/HP, Pontifícia Universidade Católica do RGS, Porto Alegre, RS, Brasil, Dez 2007.

[47] The OpenVZ Team. *VZCTL - Utility to Control a Virtual Environment*. 2007. Disponível em: <http://openvz.org/documentation/mans/vzctl.8>. Acesso em: 21 out. 2007.

[48] MELLOR, E.; SHARP, R.; SCOTT, D. *Xen Management API*. 2007. Disponível em: <http://wiki.xensource.com/xenwiki/XenApi>. Acesso em: 3 set. 2007.

A vMIB

```
--Author: Guilherme da Cunha Rodrigues.
--e-mail: grodrigues@inf.pucrs.br

VIRTUAL-MIB DEFINITIONS ::= BEGIN

IMPORTS
  experimental
      FROM RFC1155-SMI
  OBJECT-TYPE
      FROM RFC1155-SMI
  DisplayString
     FROM RFC1213-MIB;

virtual  OBJECT IDENTIFIER ::= { experimental 1 }


--TEXTUAL CONVENTIONS-
--
--      private         OBJECT IDENTIFIER
--              ::= { internet 4 }
--
--       virtual    OBJECT IDENTIFIER
--               ::= { experimental 1 }
--
--        enterprise      OBJECT IDENTIFIER
--                ::= { private 1 }
--

 physicalResources  OBJECT IDENTIFIER ::= { virtual 1 }

 virtualResources  OBJECT IDENTIFIER ::= { virtual 2 }

--
-- Physical Resources Subtree
--

--      This section describes real resources, such resources
        are the following:
--      Processor, Memory, Disk and Network.
```

```
--

--
--      Processor
--

processor  OBJECT IDENTIFIER
 ::= { physicalResources 1 }

        processorNumber OBJECT-TYPE
                SYNTAX          INTEGER
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Number of physical CPUs
                                 on the system."
                ::= { processor 1}

        processorTable OBJECT-TYPE
                SYNTAX          SEQUENCE OF ProcessorEntry
                MAX-ACCESS      not-accessible
                STATUS          current
                DESCRIPTION     "Table for physical CPUs."
                ::= { processor 2}

        processorEntry  OBJECT-TYPE
                SYNTAX          ProcessorEntry
                MAX-ACCESS      not-accessible
                STATUS          current
                DESCRIPTION     "Description of processor
                                 features"
                INDEX           { processorId }
                ::= { processorTable 1}

        ProcessorEntry ::= SEQUENCE { processorId  INTEGER,
processorCore    INTEGER,
        processorType DisplayString,
processorSpeed INTEGER,
processorFlags DisplayString,
processorVirtEnabled INTEGER}
        processorId OBJECT-TYPE
                SYNTAX          INTEGER
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Processor identification."
                ::= { processorEntry 1 }

        processorCore OBJECT-TYPE
                SYNTAX          INTEGER
```

```
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Number of cores on
                                 the processor."
                ::= { processorEntry 2 }

        processorType OBJECT-TYPE
                SYNTAX          DisplayString
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Processor type
                                 description."
                ::= { processorEntry 3 }

        processorSpeed OBJECT-TYPE
                SYNTAX          INTEGER
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Performance capability
                                 of the processor."
                ::= { processorEntry 4 }

        processorFlags OBJECT-TYPE
                SYNTAX          DisplayString
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Control Flag."
                ::= { processorEntry 5 }

        processorVirtEnabled OBJECT-TYPE
                SYNTAX          INTEGER
                       {on (1),
        off (2)}
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "It has suport to
                                 virtualization."
                ::= { processorEntry 6 }



--
--      Memory
--

    memory OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
```

```
STATUS current
DESCRIPTION "memory"
::= {physicalResources 2 }

        memoryTotal OBJECT-TYPE
                SYNTAX          INTEGER
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Amount of memory in
                                 the system."
                ::= { memory 1}

        memoryFree OBJECT-TYPE
                SYNTAX          INTEGER
                MAX-ACCESS      read-only
                STATUS          current
                DESCRIPTION     "Amount of free memory in
                                 the system."
                ::= { memory 2}

--
--      Disk
--

disk OBJECT-TYPE
            SYNTAX INTEGER
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION "Disk"
       ::= { physicalResources 3 }

diskTable OBJECT-TYPE
            SYNTAX SEQUENCE OF DiskEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION "Table for physical disk."
::={ disk 1 }

diskEntry OBJECT-TYPE
            SYNTAX DiskEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION "Description of disk features."
::= { diskTable 1 }

DiskEntry ::= SEQUENCE
{
diskIn INTEGER,
```

```
diskName DisplayString,
diskVmId INTEGER
}

diskIn OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The disk Index."
    ::={ diskEntry 1 }

diskName OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The disk Name."
    ::={ diskEntry 2 }

diskVmId OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The disk on the Vm."
    ::={ diskEntry 3 }


--
--      Network
--

network OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "Network"
      ::= { physicalResources 4 }

networkTable OBJECT-TYPE
    SYNTAX SEQUENCE OF NetworkEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION "Table for physical network interfaces."
    ::= { network 1 }

networkEntry OBJECT-TYPE
    SYNTAX NetworkEntry
    MAX-ACCESS not-accessible
    STATUS current
```

```
    DESCRIPTION "Description of network features."
    ::= { networkTable 1 }

NetworkEntry ::= SEQUENCE
{
networkIndex INTEGER,
networkName DisplayString,
networkInKbytes Counter,
networkInPackets Counter,
networkInErrors Counter,
networkInDiscards Counter,
networkOutKbytes Counter,
networkOutPackets Counter,
networkOutErrors Counter,
networkOutDiscards Counter,
networkMac DisplayString,
networkIp4 ipaddress,
networkIp6 DisplayString,
networkBro ipaddress,
networkGate DisplayString,
networkLocalNet ipaddress
}

networkIndex OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION  "Index for network table."
    ::= { networkEntry 1 }

networkName OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "Describe the network interface."
    ::= { networkEntry 2 }

networkInKbytes OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The number of Kbytes received on
             the network interface."
    ::= { networkEntry 3 }

networkInPackets OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
```

```
    STATUS current
    DESCRIPTION "The number of packets received on
             the network interface."
    ::= { networkEntry 4 }

networkInErrors OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The number of erroneous packets received
                 on the network interface."
    ::= { networkEntry 5 }

networkInDiscards OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The number of dropped packets received
    on the network interface."
    ::= { networkEntry 6 }

networkOutKbytes OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
     STATUS current
    DESCRIPTION "The number of Kbytes sent on
             the network interface."
    ::= { networkEntry 7 }

networkOutPackets OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The number of packets sent on
             the network interface."
    ::= { networkEntry 8 }

networkOutErrors OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The number of packets that could not be sent
                 on the network interface because any errors."
    ::= { networkEntry 9 }

networkOutDiscards OBJECT-TYPE
    SYNTAX Counter
    MAX-ACCESS read-only
```

```
    STATUS current
    DESCRIPTION "The number of packets that have not been sent
                 on the network interface even though no
                 errors."
    ::= { networkEntry 10 }

networkMac OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The MAC Address on the network interface."
     ::= { networkEntry 11 }

networkIp4 OBJECT-TYPE
    SYNTAX ipaddress
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The IP number for IPv4 version."
    ::={ networkEntry 12 }

networkIp6 OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The IP number for IPv6 version."
    ::={ networkEntry 13 }

networkBro OBJECT-TYPE
    SYNTAX ipaddress
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION " The IP number for broadcast."
    ::={ networkEntry 14 }

networkGate OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The network gateway."
    ::={ networkEntry 15 }

networkLocalNet OBJECT-TYPE
    SYNTAX ipaddress
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION "The local network."
    ::={ networkEntry 16}
```

```
--
--      Virtual Resources Subtree
--

--
--      This section describes virtual resources,
        such resources are the following:
--      Processor, memory, Disk , Network, Virtual
        Machine and Virtual Machine Monitor (VMM).
--



--
--      Virtual Machine
--

  virtualMachine  OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Virtual Machine."
            ::= { virtualResources 1}


  virtualMachineTable     OBJECT-TYPE
        SYNTAX          SEQUENCE OF VirtualMachineEntry
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Table for virtual machines(VMs)
                         status."
        ::= { virtualMachine 1 }

 virtualMachineEntry OBJECT-TYPE
        SYNTAX VirtualMachineEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Description of virtual machines features."
        INDEX {virtualVmIdVm }
        ::= { virtualMachineTable 1}

        VirtualMachineEntry       ::= SEQUENCE
{
      virtualVmIdVm INTEGER,
      virtualMachineName DisplayString,
      virtualMachineStatus INTEGER
}
 virtualVmIdVm     OBJECT-TYPE
```

```
       SYNTAX          INTEGER
       MAX-ACCESS      read-only
       STATUS          current
       DESCRIPTION     "Identification of the
                          virtual machine (VM)."
       ::= { virtualMachineEntry 1}

--
--      Virtual Processor
--

 virtualProcessor OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Virtual Processor."
            ::= { virtualVmIdVm 1 }

 virtualProcessorNumber   OBJECT-TYPE
        SYNTAX          INTEGER
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Number of virtual processors
                         in the system."
        ::= { virtualProcessor 1 }

 virtualProcessorTable    OBJECT-TYPE
        SYNTAX          SEQUENCE OF VirtualProcessorEntry
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Table for virtual CPUs."
        ::= { virtualProcessor 2 }

 virtualProcessorEntry OBJECT-TYPE
        SYNTAX VirtualProcessorEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Description of virtual processors
                     features."
        INDEX { virtualVmId }
        ::= { virtualProcessorTable 1 }

 VirtualProcessorEntry  ::= SEQUENCE
{
        virtualVmId     INTEGER,
        virtualProcessorId      INTEGER
}
 virtualVmId   OBJECT-TYPE
```

```
        SYNTAX          INTEGER
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Identification of the
                         Virtual Machine (VM)."
        ::= { virtualProcessorEntry 1}

 virtualProcessorId      OBJECT-TYPE
        SYNTAX          INTEGER
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Identification of the processor
                         on the Virtual Machine (VM)."
        ::= { virtualProcessorEntry 2}

--
--      Virtual Memory
--

 virtualMemory    OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Virtual memory."
            ::= { virtualVmIdVm 2}

 virtualMemoryTable      OBJECT-TYPE
        SYNTAX          SEQUENCE OF VirtualProcessorEntry
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Table for virtual CPUs."
        ::= { virtualMemory 1}

 virtualMemoryEntry OBJECT-TYPE
        SYNTAX VirtualMemoryEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Description of virtual memory features."
        INDEX { virtualVmIdM }
        ::= { virtualMemoryTable 1 }

 VirtualMemoryEntry  ::= SEQUENCE{virtualVmIdM INTEGER,
 virtualMemoryTotal      INTEGER,
 virtualMemoryUsed       INTEGER}

 virtualVmIdM OBJECT-TYPE
        SYNTAX          INTEGER
        MAX-ACCESS      read-only
```

```
        STATUS          current
        DESCRIPTION     "Identification of the
                         Virtual Machine (VM)."
        ::= { virtualMemoryEntry 1}

 virtualMemoryTotal      OBJECT-TYPE
        SYNTAX          INTEGER
        MAX-ACCESS      read-write
        STATUS          current
        DESCRIPTION     "Amount of the memory alocate
                         in the Virtual Machine (VM)."
        ::= { virtualMemoryEntry 2}

 virtualMemoryUsed       OBJECT-TYPE
        SYNTAX          INTEGER
        MAX-ACCESS      read-write
        STATUS          current
        DESCRIPTION     "Amount of the memory used in
                         the Virtual Machine (VM)."
        ::= { virtualMemoryEntry 3}


--
--      Virtual Disk
--

 virtualDisk OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Virtual Disk."
          ::= { virtualVmIdVm 3 }

 virtualDiskTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VirtualDiskEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Table for virtual disk."
        ::={ virtualDisk 1 }

 virtualDiskEntry OBJECT-TYPE
        SYNTAX VirtualDiskEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION "Description of disk features."
        ::= { virtualDiskTable 1 }

VirtualDiskEntry ::= SEQUENCE
```

```
{
virtualDiskId INTEGER,
virtualDiskName DisplayString,
virtualDiskVmId INTEGER
}

 virtualDiskId OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "The virtual disk Id."
       ::={ virtualDiskEntry 1 }

 virtualDiskName OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "The virtual disk Name."
       ::={ virtualDiskEntry 2 }

 virtualDiskVmId OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "The virtual disk on the Vm."
       ::={ virtualDiskEntry 3 }

--
--      Virtual Network
--

 virtualNetwork OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "Virtual Network."
         ::= { virtualVmIdVm 4 }

 virtualNetworkTable OBJECT-TYPE
       SYNTAX SEQUENCE OF VirtualNetworkEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION "Table for virtual network interfaces."
       ::= { virtualNetwork 1 }

 virtualNetworkEntry OBJECT-TYPE
       SYNTAX VirtualNetworkEntry
       MAX-ACCESS not-accessible
```

```
       STATUS current
       DESCRIPTION "Description of network features."
       ::= { virtualNetworkTable 1 }

VirtualNetworkEntry ::= SEQUENCE
{
virtualNetworkIndex INTEGER,
virtualNetworkName DisplayString,
virtualNetworkInKbytes Counter,
virtualNetworkInPackets Counter,
virtualNetworkInErrors Counter,
virtualNetworkInDiscards Counter,
virtualNetworkOutKbytes Counter,
virtualNetworkOutPackets Counter,
virtualNetworkOutErrors Counter,
virtualNetworkOutDiscards Counter,
virtualVmIdN INTEGER,
virtualNetworkMac DisplayString,
virtualNetworkIp4 ipaddress,
virtualNetworkIp6 DisplayString,
virtualNetworkBro ipaddress,
virtualNetworkGate DisplayString,
virtualNetworkLocalNet ipaddress
}

 virtualNetworkIndex OBJECT-TYPE
       SYNTAX INTEGER
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION  "Index for virtual network table."
       ::= { virtualNetworkEntry 1 }

 virtualNetworkName OBJECT-TYPE
       SYNTAX DisplayString
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "Describe the virtual network interface."
       ::= { virtualNetworkEntry 2 }

 virtualNetworkInKbytes OBJECT-TYPE
       SYNTAX Counter
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION "The number of Kbytes received on
                    the virtual network interface."
       ::= { virtualNetworkEntry 3 }

 virtualNetworkInPackets OBJECT-TYPE
```

```
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of packets received on
                   the virtual network interface."
      ::= { virtualNetworkEntry 4 }

virtualNetworkInErrors OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of erroneous packets received on
                   the virtual network interface."
      ::= { virtualNetworkEntry 5 }

virtualNetworkInDiscards  OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of dropped packets received on
                   the virtual network interface."
      ::= { virtualNetworkEntry 6 }

virtualNetworkOutKbytes OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of Kbytes sent on
                   the network virtual interface."
      ::= { virtualNetworkEntry 7 }

virtualNetworkOutPackets OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of packets sent on
                   the network virtual interface."
      ::= { virtualNetworkEntry 8 }

virtualNetworkOutErrors OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of packets that could not
                   be sent on the virtual network interface
                   because any errors."
      ::= { virtualNetworkEntry 9 }
```

```
virtualNetworkOutDiscards OBJECT-TYPE
      SYNTAX Counter
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The number of packets that have not been
                   sent on the virtual network interface
                   even though no errors."
      ::= { virtualNetworkEntry 10 }

virtualVmIdN OBJECT-TYPE
      SYNTAX INTEGER
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "ID of the VM on virtual network."
      ::= { virtualNetworkEntry 11 }

virtualNetworkMac OBJECT-TYPE
      SYNTAX DisplayString
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The virtual MAC Address on the
                   network interface."
      ::= { virtualNetworkEntry 12 }

virtualNetworkIp4 OBJECT-TYPE
      SYNTAX ipaddress
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The virtual IP number for IPv4 version."
      ::={ virtualNetworkEntry 13 }

virtualNetworkIp6 OBJECT-TYPE
      SYNTAX DisplayString
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION "The virtual IP number for IPv6 version."
      ::={ virtualNetworkEntry 14 }

virtualNetworkBro OBJECT-TYPE
      SYNTAX ipaddress
      MAX-ACCESS read-only
      STATUS current
      DESCRIPTION " The virtual IP number for broadcast."
      ::={ virtualNetworkEntry 15 }

virtualNetworkGate OBJECT-TYPE
      SYNTAX DisplayString
      MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION "The virtual network gateway."
        ::={ virtualNetworkEntry 16 }

    virtualNetworkLocalNet OBJECT-TYPE
        SYNTAX ipaddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "The local virtual network."
        ::={ virtualNetworkEntry 17 }


    virtualMachineName      OBJECT-TYPE
        SYNTAX          DisplayString
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "Name of the
                         virtual machine (VM)."
        ::= { virtualMachineEntry 2}


    virtualMachineStatus OBJECT-TYPE
        SYNTAX DisplayString{
                         running (1),
                         blocked (2),
                         paused (3),
                         shutdown(4)}
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Currenly status of virtual machine."
        ::= { virtualMachineEntry 3}


--
--      Virtual Machine Monitor
--

    virtualMachineMonitor  OBJECT-TYPE
        SYNTAX INTEGER
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "Virtual machine monitor."
        ::= { virtualResources 2}

    virtualMachineMonitorSystem     OBJECT-TYPE
        SYNTAX          DisplayString
        MAX-ACCESS      read-only
        STATUS          current
```

```
       DESCRIPTION     "System Manager of
                        virtual machine (VM)."
         ::= { virtualMachineMonitor 1}

 virtualMachineMonitorVersion    OBJECT-TYPE
         SYNTAX          INTEGER
         MAX-ACCESS      read-only
         STATUS          current
         DESCRIPTION     "System Manager version of
                          virtual machine (VM)."
         ::= { virtualMachineMonitor 2}

 virtualMachineMonitorBuild      OBJECT-TYPE
         SYNTAX          INTEGER
         MAX-ACCESS      read-only
         STATUS          current
         DESCRIPTION     "System Manager created of
                          virtual machine (VM)."
         ::= { virtualMachineMonitor 3}

END
```