UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FÁBIO FABIAN DAITX

Uma Solução Baseada em SNMP para Gerenciamento de Dispositivos de Rede com Suporte à Virtualização

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Lisandro Z. Granville
Orientador

Porto Alegre, Agosto de 2011

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Daitx, Fábio Fabian

Uma Solução Baseada em SNMP para Gerenciamento de Dispositivos de Rede com Suporte à Virtualização / Fábio Fabian Daitx. – Porto Alegre: PPGC da UFRGS, 2011.

91 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Orientador: Lisandro Z. Granville.

1. Roteador Virtual. 2. Virtualização de Rede. 3. SNMP (*Simple Network Management Protocol*). 4. MIB (*Management Information Base*). I. Granville, Lisandro Z.. II. Título.

"Os dez mandamentos do sucesso são:

1. Trabalhar duro: trabalhar duro é o melhor investimento que um homem pode fazer.

2. Estudar com afinco: o conhecimento permite a um homem trabalhar de forma mais inteligente e eficaz.

3. Ter iniciativa: a rotina geralmente termina no túmulo.

4. Amar o seu trabalho: assim, você encontrará prazer em realizá-lo.

5. Ser exato: métodos inexatos levam a resultados inexatos.

6. Ter o espírito da conquista: assim, você pode lutar com sucesso e vencer as dificuldades.

7. Cultivar a personalidade: a personalidade é para o homem o que o perfume é para as flores.

8. Ajudar e dividir com os demais: o teste real da grandeza de um negócio está em criar oportunidade para outras pessoas.

9. Ser democrático: a menos que você seja correto com seus companheiros, você nunca poderá ser um líder de sucesso.

10. Dar o melhor de si em todos os momentos: a pessoa que deu o melhor de si fez tudo. A pessoa que não deu todo o seu melhor, não fez nada."

— Charles M. Schwab

AGRADECIMENTOS

Primeiramente, dirijo meus agradecimentos ao professor Lisandro Zambenedetti Granville, meu orientador, por ter apostado em mim, oportunizando a realização deste trabalho. Ele se manteve presente, mesmo com a distância, dando o apoio necessário e decisivo para as atividades desenvolvidas.

Em segundo lugar, mas não menos importante, gostaria de agradecer ao colega Rafael Esteves, por ter participado diretamente neste trabalho, com valiosas opiniões, sugestões e apoio nas implementações e testes. Rafael, muito obrigado pela ajuda e sucesso em seu Doutorado.

Gostaria também de agradecer aos membros do Grupo de Redes da UFRGS, pela experiência e convívio que tivemos ao longo do ano de 2009. Weverton Cordeiro, Bruno Dalmazo, Rodrigo Mansilha, Juliano Wickboldt, Ricardo Luis, Roben Lunardi, Flávio Santos, Alessandro Santos, prof. Luciano Paschoal e demais participantes, parabéns pelo excelente trabalho realizado aí na UFRGS. Vocês fazem parte de um grupo seleto de pesquisadores e certamente me ajudaram de alguma forma a concluir essa dissertação.

Por fim, dedico este trabalho a minha família, minha mãe Elia Fabian Daitx, meu pai Adão de Oliveira Daitx e meu irmão Jean Fabian Daitx. Agradeço a eles, por terem me passado ensinamentos valiosos ao longo de minha vida e por terem apoiado minhas decisões profissionais, celebrando sempre a meu lado as conquistas alcançadas. À minha namorada, Josiane de Lima Scherer, agradeço pela compreensão e carinho nestes últimos 2 anos de minha vida.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ASON	Automatically Switched Optical Network
ATM	Asynchronous Transfer Mode
BoD	Bandwidth on Demand
BGP	Border Gateway Protocol
CE	Customer Edge
CLI	Command Line Interface
DB	Data Base
DNS	Domain Name Server
FE	Fast Ethernet
FRR	Fast Re-Route
GE	Gigabit Ethernet
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
InP	Infrastructure Provider
IOS	Internetwork Operating System
IP	Internet Protocol
ISP	Internet Service Provider
IT	Information Technology
ITIL	Information Technology Infrastructure Library
L1VPN	Layer 1 Virtual Private Network
L3 PPVPN	Layer 3 Provider Provisioned Virtual Private Network
LR	Logical Router
MIB	Management Information Base
MPLS	Multiprotocol Label Switching

NAT	Network Address Translation
NIA	Network Interconnection Agreement
NIC	Network Interface Card
NP	Network Plane
NP-Hard	Non-deterministic Polynomial-time Hard
NVE	Network Virtualization Environment
OS	Operating System
OSPF	Open Shortest Path First
P	Provider
PDU	Protocol Data Unit
PE	Provider Edge
POP	Point of Presence
RFC	Request for Comments
RIP	Routing Information Protocol
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SP	Service Provider
STM	Synchronous Transport Module
URL	Uniform Resource Locator
VI	Virtual Interface
VLAN	Virtual Local Area Network
VN	Virtual Network
VPLS	Virtual Private LAN Service
VR	Virtual Router
VPN	Virtual Private Network
XML	Extended Markup Language
WAN	Wide Area Network

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A virtualização de rede surgiu como uma alternativa para contornar as limitações no uso compartilhado da infra-estrutura atual da Internet. Com a virtualização, sob uma mesma estrutura física, ou substrato, é possível a construção de múltiplas redes virtuais, cada uma das quais empregando seus próprios protocolos, mecanismos de endereçamento e políticas de forma independente e isolada. Essas redes são formadas por roteadores, interfaces e enlaces virtuais mapeados sobre componentes reais. Por serem desacopláveis, os elementos virtuais trazem uma grande flexibilidade, sendo possível a construção de múltiplas redes sobrepostas para usuários com demandas distintas, por exemplo. Roteadores virtuais podem migrar de um roteador físico para outro, conforme necessidades de manutenção ou balanceamento de carga. Além disso, a possibilidade de se conduzir experimentos de rede em uma estrutura real, sujeita às condições de utilização e de tráfego que normalmente serão encontradas na prática, sem que se precise interromper a operação da rede, traz a possibilidade de se instalar poderosos e sofisticados ambientes de teste (*testbeds*).

Nesse ambiente, contudo, existem desafios de pesquisa a serem tratados e questões em aberto, em especial com relação ao gerenciamento de dispositivos. Em uma rede com suporte a virtualização, os roteadores físicos precisam ser gerenciados para que roteadores virtuais possam ser criados, modificados, duplicados, destruídos e movimentados. As interfaces de gerenciamento atuais, porém, não suportam tais ações de forma efetiva, obrigando o administrador dos dispositivos a realizar intervenções manuais através de interfaces de linha de comandos (CLIs) não padronizadas. Existe, assim, a necessidade de se definir uma interface de gerenciamento adequada para roteadores físicos que abrigam roteadores virtuais.

Tendo como objetivo abordar estas questões, este trabalho consiste na investigação de uma solução para gerenciar roteadores virtuais. Para isto foram levantadas e definidas as principais ações de gerenciamento necessárias aos dispositivos, de forma integrada, visando a definição de uma interface padronizada de gerenciamento para cenários heterogêneos de utilização.

Palavras-chave: Roteador Virtual, Virtualização de Rede, SNMP (*Simple Network Management Protocol*), MIB (*Management Information Base*).

A Solution Based on SNMP for Management of Network Devices Supporting Virtualization

ABSTRACT

Network virtualization emerged as an alternative to overcome limitations on use of the shared infrastructure of current Internet. With virtualization, over the same physical structure, or substrate, it is possible to build multiple virtual networks, each of them employing its own protocols, addressing mechanisms and policies, on an isolated and independent way. These networks are formed by virtual routers, interfaces and links mapped to real components. Because they are detachable, the virtual elements bring a great flexibility, being possible constructing many overlaid networks for users with different demands, for example. Virtual routers can migrate from one physical router to another as needed form maintenance or load balancing. Besides, the possibily of running network experiments in a real structure, subject to the conditions of use and traffic that will normally be encountered in practice, without interrupting network operation, make it possible to install powerfull and sofisticated testbeds.

In such environment, however, there are research challenges to be managed and open questions, specially related to devices management. In a network with virtualization support, physicall routers must be managed so that virtual routers can be created, modified, copied, removed and moved. Nevertheless, current management interfaces do not support such action in an effictive way, compelling the devices manager to make manual interventions through non standardized command line interfaces (CLIs). So, there exist the need to define an adequate management interface for physicall routers that host virtual routers.

Having as an objective to address these questions, this work consist in investigating a solution to manage virtual routers. For this, were raised and defined the main management actions required for devices, through an integrated way, aiming to define a standardized management interface for heterogeneous use scenarios.

Keywords: Virtual Router, Network Virtualization, SNMP (Simple Network Management Protocol), MIB (Management Information Base).

1 INTRODUÇÃO

A Internet vem experimentando uma evolução significativa ao longo de sua história. A estrutura originalmente simples de seu modelo baseado na oferta de um serviço de melhor esforço, no núcleo da rede, não comporta mais a demanda por novos serviços e a explosão no volume de tráfego global de informações. O acréscimo contínuo de novas funcionalidades contribuiu para o aumento da complexidade da rede, tornando-se cada vez mais difícil a implementação de novos modelos, protocolos e serviços na estrutura atual (TAYLOR; TURNER, 2004) e culminou naquilo que vem sendo definido por muitos pesquisadores como a "ossificação" da Internet (ANDERSON et al., 2005).

A virtualização de redes (RIXNER, 2008) surgiu como uma alternativa para impulsionar o desenvolvimento de novas arquiteturas de redes e para com isso auxiliar a superar a ossificação da Internet (TURNER; MCKEOWN, 2007). Na virtualização de redes, a infraestrutura física, chamada substrato, é compartilhada entre múltiplas redes virtuais, cada uma das quais empregando seus próprios protocolos, esquemas de endereçamento e políticas de forma independente e isolada.

De uma forma geral, o advento da virtualização e sua disseminação são fenômenos naturais decorrentes da crescente utilização de arquiteturas com múltiplas unidades de processamento. Normalmente, o emprego desta tecnologia traz significativos resultados de economia energética, financeira e de utilização de espaço físico. Aplicada às redes, a virtualização flexibiliza o gerenciamento dos dispositivos através da redução no acoplamento *hardawer-software*. Isso traz a ideia de que os equipamentos (dispositivos físicos) podem ser alugados em fatias e que a infraestrutura da rede, em geral, se torna um serviço (IaaS, ou *Infrastructure as a Service*). Além disso, novas funcionalidades importantes podem ser implementadas, como a migração de equipamentos virtuais, que reduz os tempos de indisponibilidade da rede em casos de falha ou manutenção.

Os roteadores congregam grande quantidade de funcionalidades associadas à inteligênica no processamento de pacotes, sendo estas de vital importância para atuação dos provedores de serviços. Embora a pesquisa em virtualização de rede seja bastante ativa atualmente, poucos estudos são encontrados no que tange o gerenciamento dos roteadores físicos que são capazes de hospedar roteadores virtuais. Na prática, a maioria das ações de gerenciamento sobre esses dispositivos requer interevenção manual via interfaces por linha de comando (*command line interface*, ou CLI). Apesar de suficiente para administração dos atuais substratos de rede em pequena escala, isso certamente não será suficiente para o gerenciamento de ambientes na Futura Internet.

Além das limitações existentes nas propostas supracitadas, alguns desafios a

serem vencidos para o desenvolvimento de uma solução de gerenciamento para roteadores virtuais permanecem sem solução:

i) **Requisitos de gerenciamento de uma interface padrão para roteadores físicos e virtuais:** deve ser apresentada uma forma organizada e objetiva para acesso às funcionalidades e informações dos roteadores, que permita padronizar o acesso aos recursos de diferentes plataformas de virtualização, abstraindo os aspectos de implementação das mesmas;

ii) **Principais parâmetros de desempenho a serem monitorados:** o desempenho do *hardware* deve ser monitorado e controlado, visto que o mesmo pode impactar diretamente nos protocolos e funcionalidades desempenhadas pelo dispositivo, ou mesmo no tráfego dos usuários roteado pelo equipamento;

iii) **Formas de endereçamento de elementos físicos e virtuais:** com a virtualização, o roteador físico é expandido em múltiplos roteadores virtuais. As interfaces físicas são igualmente expandidas e compartilhadas por múltiplas interfaces virtuais. Dessa forma, o endereçamento destes componentes e o esquema de gerenciamento precisam ser definidos e organizados;

iv) **Identificação das principais ações de gerenciamento:** é necessário reconhecer o conjunto de operações fundamentais (ou "primitivas") que precisam ser implementadas e gerenciadas no contexto de operacionalização das redes virtuais, especialmente aquelas aplicadas ao gerenciamento de configuração dos dispositivos;

v) **Comparação de desempenho do gerenciamento de diferentes plataformas de virtualização:** é indispensável avaliar o impacto que o *hardware* pode ter no gerenciamento e encontrar formas para fazer este gerenciamento se tornar o menos intrusivo possível aos dispositivos. Devem ser identificados os recursos disponíveis em cada plataforma e meios de mapea-los para uma interface de gerenciamento padrão.

Roteadores virtuais (ou roteadores lógicos) são implementações por *software* de roteadores reais isolados sobre o mesmo dispositivo físico. Estes elementos são considerados essenciais para a "virtualização de rede" e podem ser implementados por plataformas de "virtualização de nodos". Ambientes completos de rede podem ser criados com o uso de tecnologias como VMware (VMWARE, 1998) e Xen (XEN, 2003). Máquinas virtuais e roteadores virtuais podem ser utilizados para formar tais ambientes nas pontas e no núcleo da rede, respectivamente. Como na virtualização de servidores, a virtualização de rede cria uma estrutura mais flexível, onde, por exemplo, um roteador virtual pode migrar de um roteador físico para outro. Essa flexibilidade leva à noção de que esses elementos serão componentes essenciais na Futura Internet (PAPADIMITRIOU et al., 2009).

Esta dissertação de mestrado tem por objetivo propor uma solução de gerenciamento para roteadores virtuais que permita monitorar, configurar e operar esses dispositivos pela rede, por meio de um protocolo de gerenciamento. Para isso, esta dissertação apresenta uma interface de gerenciamento que contempla as ações necessárias na manipulação desses recursos, bem como um agente que implementa o acesso às funcionalidades dos *hardwares* e *softwares* estudados. A fim de avaliar

a solução proposta, foram realizados experimentos, e são apresentados os resultados de desempenho, identificando os principais pontos críticos e limitações a serem contornadas.

O restante desta dissertação está organizado como segue. O Capítulo 2 introduz alguns dos conceitos que formam a base da investigação apresentada doravante e retoma o trabalho relacionado. No Capítulo 3 é apresentada a solução proposta neste trabalho e são estudados cenários de gerenciamento a fim de avaliar a aplicabilidade e os requisitos para a interface desenvolvida. O Capítulo 4 discute os resultados obtidos com a avaliação experimental. Por fim, o Capítulo 5 concluí esta dissertação com as considerações finais e perspectivas sobre direções de pesquisa e tendências nesta área.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos chave sobre virtualização de redes (Seção 2.1), modelo de arquitetura de redes virtuais (Seção 2.2) e virtualização de roteadores (Seção 2.3). São também discutidos os trabalhos de pesquisa mais importantes relacionados a esta dissertação (Seção 2.4).

2.1 Virtualização de Redes

O termo "virtualização de rede" é muitas vezes utilizado para se referenciar diferentes assuntos. "Virtualização de rede", estritamente, geralmente se refere a um ambiente de rede onde os provedores de serviços (*Internet Service Providers*, ou ISPs) dividem seus papéis entre provedores de infraestrutura (*Infrastructure Providers*, ou InPs) e provedores de serviços (*Service Providers*, ou SPs). Os provedores de serviços (SPs) agregam recursos de múltiplos provedores de infraestrutura (InPs), a fim de fornecer serviços fim-a-fim (Seção 2.2). Algumas vezes, esse conceito também é chamado de "infraestrutura como um serviço" (*Infrastructure as a Service*, ou IaaS) (LEMAY, 2008). A arquitetura de uma Internet virtual é formada por hosts virtuais, roteadores virtuais e enlaces virtuais (TOUCH et al., 2003).

2.1.1 Roteador Virtual

Roteador Virtual (VR, ou *Virtual Router*) é um equipamento que pode hospedar elementos virtuais, emulando a funcionalidade de múltiplos roteadores reais.

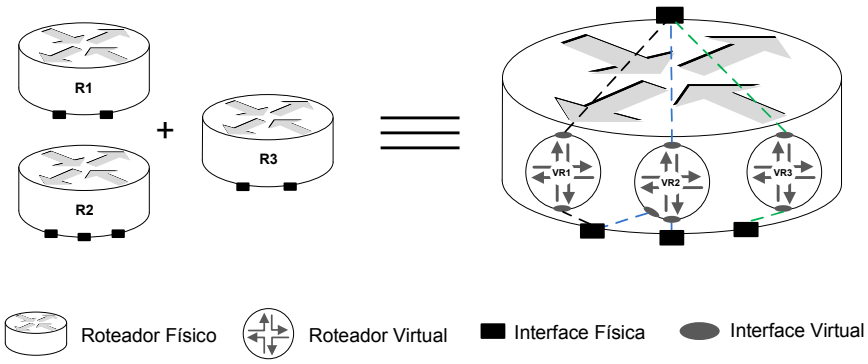


Figura 2.1: Representação de um Roteador Virtual.

O roteador virtual, mais especificamente, é o aplicativo roteador (similar a uma máquina virtual) que roda em cima do equipamento (roteador físico) com capacidade para abrigar os VRs. A Figura 2.1 mostra a representação de três roteadores físicos,

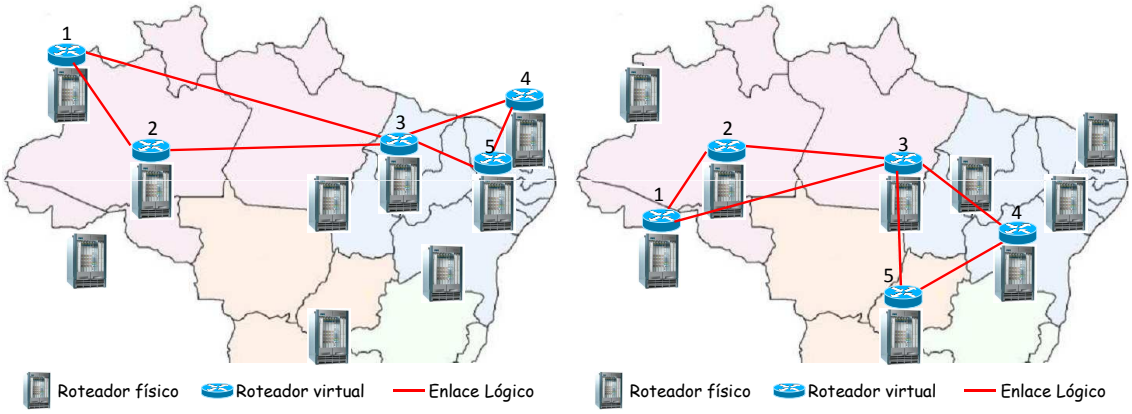
R1, R2 e R3 (à esquerda), e seu equivalente em um único dispositivo que hospeda três roteadores virtuais, VR1, VR2 e VR3 (à direita). Neste exemplo, as interfaces físicas dos roteadores R1, R2 e R3 foram transformadas em interfaces virtuais dos roteadores virtuais VR1, VR2 e VR3 e associadas a quatro interfaces reais do novo equipamento.

O roteador virtual é baseado no principio da virtualização de nodos da rede, que consiste em fazer com que um único dispositivo físico se comporte como (ou se pareça com) múltiplas instâncias de um dispositivo real (MAIER; HERRSCHER; ROTHERMEL, 2007). Nodos virtuais são dispositivos lógicos (*software*) que são executados sobre dispositivos físicos (*hardware*).

Roteadores virtuais são utilizados desde o começo dos anos 2000 para implementar redes privadas virtuais de nível 3 fornecidas pelos provedores (*Layer 3 Provider Provisioned Virtual Private Networks*, ou L3 PPVPN) para comunicações entre localidades (*site-to-site*). Nessa abordagem, o dispositivo de borda do provedor (*provider edge*, ou PE) contém um roteador virtual por VPN (com domínios de roteamento logicamente independentes), não sendo necessários mecanismos extra de endereçamento para alcançar as VPNs (KNIGHT et al., 2003). Uma implementação de *backbone* utilizando essa estratégia foi instalada na rede SINET3 (URUSHIDANI et al., 2008).

2.1.2 Virtualização de Enlace

O conceito de topologia lógica é essencial para entender a virtualização de enlaces. Topologias lógicas formam redes virtuais.



(a) Topologia lógica hospedada sobre um subtrato (conjunto de roteadores físicos).

(b) Mesma topologia lógica sobre um substrato diferente (conjunto de roteadores físicos diferente).

Figura 2.2: Exemplo de topologias lógicas idênticas hospedadas de duas formas diferentes.

As Figuras 2.2 (a) e (b) mostram a mesma topologia lógica sobre dispositivos físicos diferentes. Nesta figura, não são apresentados os enlaces reais (físicos), apenas os virtuais. Enlaces virtuais são conexões lógicas entre dispositivos (nodos) virtuais. Embora similar, este conceito não deve ser confundido com circuitos virtuais (e tecnologias associadas tais como *Frame Relay*, ATM, MPLS e VPN, por exemplo)

que são conexões lógicas fim-a-fim entre dispositivos físicos. As topologias lógicas da Figura 2.2 são idênticas por estarem interconectando os mesmos dispositivos virtuais. Embora os nodos virtuais sejam facilmente desacopláveis dos nodos reais, os enlaces virtuais e físicos não são, como será visto na Seção 3.1.6.

2.1.3 Virtualização de Host

Hosts são, tradicionalmente, nodos fonte ou destino de dados, frequentemente dotados de recursos computacionais (memória, processamento e armazenamento). Comumente, os pacotes de dados que são recebidos por um *host*, cujo destino não seja o próprio *host*, não são encaminhados para outros locais (como em um *gateway* de rede), mesmo que o *host* possua entradas em sua tabela de roteamento IP identificando uma rota para aquele pacote. Em alguns casos, no entanto, os hosts podem ser utilizados para rotear pacotes, formando as chamadas redes em *"Overlay"*, bastante utilizadas em aplicações *peer-to-peer*, por exemplo.

As soluções de virtualização para *hosts*, tipicamente, estão disponíveis para computadores pessoais ou servidores e são implementadas por um *software* do tipo *middleware* que controla a virtualização, chamado *Hypervisor* ou *Virtual Machine Monitor (VMM)*. Dependendo da forma como este *software* está implementado e opera, a virtualização pode ser classificada, resumidamente, em:

- "Virtualização Completa" (*Full Virtualization*) também chamada "virtualização hospedada" (*hosted*), na qual o *hypervisor* é executado dentro do ambiente de um sistema operacional convencional. Neste caso, o *hypervisor* possui recursos suficientes para execução de sistemas operacionais cliente (*"guest OS"*) sem quaisquer modificações ou adaptações em seu conjunto de instruções. Exemplos de plataformas deste tipo incluem VMware Server (VMSERVER, 2009), VMware Workstation (VMWORKSTATION, 2010), Parallels Workstation (PARALLELS, 2009), Oracle VirtualBox (VIRTUALBOX, 2010) e Windows Virtual PC (VIRTUALPC, 2010);

- "Paravirtualização", também chamada "Virtualização Nativa" ou "metal à mostra" (*bare metal*), na qual os *hypervisors* são executados diretamente sobre o *hardware* do *host*. Neste caso, a máquina virtual oferece uma API especial com modificações (ou adaptações) no conjunto de instruções do sistema operacional cliente. Este tipo de virtualização, normalmente depende de suporte arquitetural em *hardware*. Exemplos de plataformas deste tipo incluem VMware ESXi/ESX (VMWAREESX, 2010), Citrix XenServer (XEN, 2003) e KVM (*Kernel-based Virtual Machine*) (KVM, 2010).

Nesta dissertação foram utilizados *hosts* finais para emular roteadores (nodos intermediários) com suporte a virtualização (Capítulo 4). Esta opção se deve ao potencial de personalização destas plataformas, que traz facilidades para projetar e avaliar a solução de gerenciamento proposta, por meio de opções de configuração flexíveis e da possibilidade de programar novas funcionalidades (Seção 2.4.3). Não é objetivo deste trabalho criar redes de *overlay* ou otimizar o desempenho dos dispositivos, mas sim obter e estudar parâmetros mensuráveis que permitam analisar quantitativamente e qualitativamente a solução de gerenciamento proposta. Seria economicamente e tecnicamente inviável trabalhar com roteadores reais, projetados exclusivamente para o processamento de pacotes, visto que estes dispositivos operam com *hardware* e *software* proprietário de alto valor comercial.

2.2 Arquitetura Conceitual de Redes Virtuais

Os conceitos básicos sobre virtualização, em especial com relação aos roteadores virtuais, foram apresentados na Seção 2.1. Nesta seção é fornecido uma visão geral da utilização destes componentes.

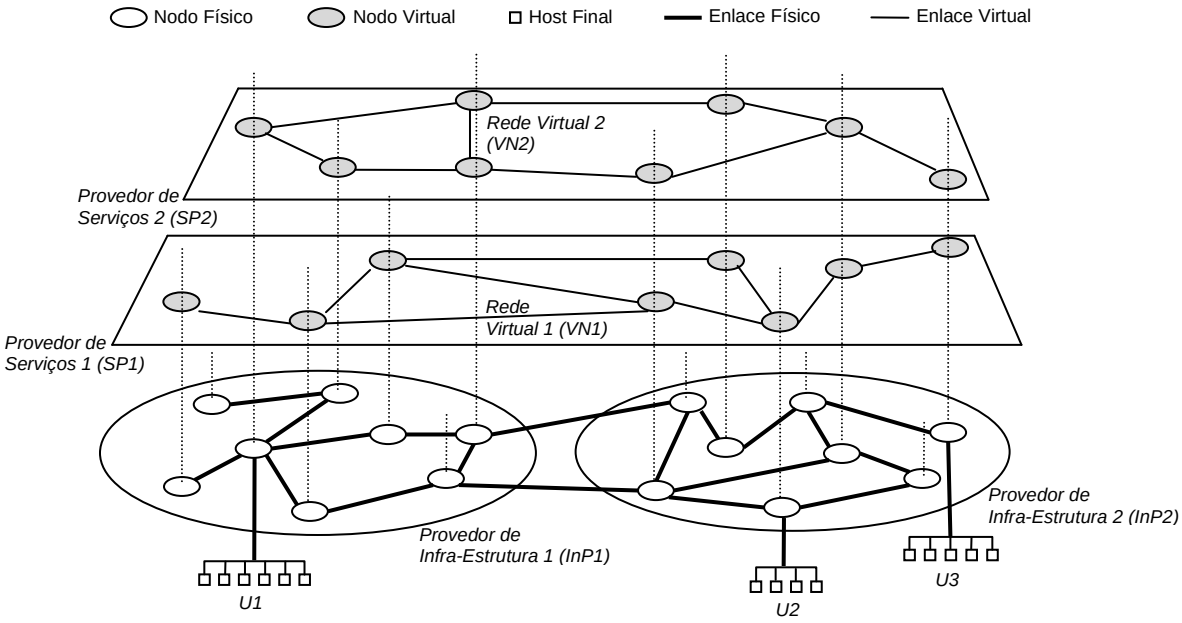
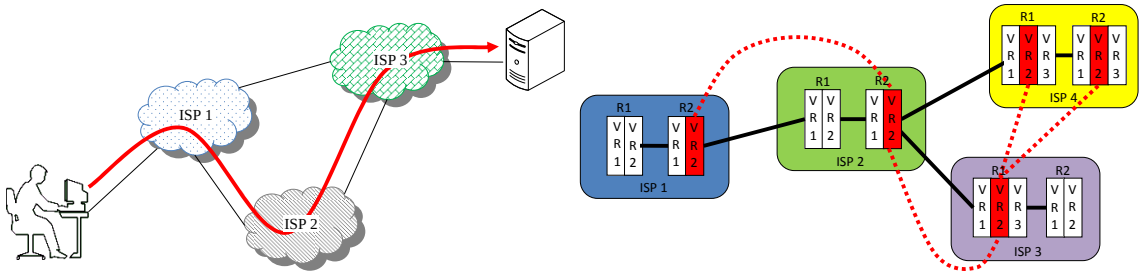


Figura 2.3: Arquitetura Típica de Redes Virtuais (CHOWDHURY; BOUTABA, 2009).

A Figura 2.3 (CHOWDHURY; BOUTABA, 2009) apresenta a arquitetura típica encontrada em redes virtuais. Neste exemplo, as redes virtuais 1 e 2 (VN1 e VN2) foram criadas pelos provedores de serviços 1 e 2 (SP1 e SP2), respectivamente. O SP1 criou sua rede VN1 sobre os recursos físicos de dois provedores de infraestrutura diferentes (InP1 e InP2), fornecendo conectividade fim-a-fim para os usuários U2 e U3. Nesta rede virtual, o usuário U1 não está conectado a estes assinantes. Já na rede VN2, do provedor SP2, os usuários U1 e U3 estão conectados (e U2 não está diretamente ligado a estes). Portanto, o usuário U3 está contratando o serviço de conectividade de ambas as redes virtuais que são gerenciadas por dois provedores de serviços distintos, que por sua vez subcontratam a infraestrutura de dois outros provedores de infraestrutura.

A virtualização de redes consiste em reunir múltiplas arquiteturas heterogêneas de redes, geralmente de diferentes provedores de serviços. Combinando o substrato de *hardware* com uma camada de *software* se cria uma visão de novas redes virtuais para o operador ou administrador. Este operador enxerga um conjunto redes de interconexões de equipamentos diferente do que realmente está implementado no baixo nível - Figura 2.3). Isso viabiliza a implantação de mais serviços e funcionalidades, otimizando a utilização dos recursos físicos da rede.

Atualmente, provedores de serviços (ISPs) de domínios autônomos diferentes fornecem a conectividade para seus usuários através do encaminhamento de pacotes entre esses domínios, sem ter uma visão do que ocorre com os mesmos nos domínios vizinhos (Figura 2.4 (a)). A solução de virtualização de redes envolve a separação entre *Service Providers* (que alugariam fatias de recursos fim-a-fim) e *Infrastructure Providers* (que forneceriam esses recursos - Figura 2.4 (b)).



(a) ISPs competidores com diferentes objetivos devem se coordenar.

(b) Um único ISP controla o caminho fim a fim.

Figura 2.4: Virtualização de redes para serviços fim-a-fim (NICK FEAMSTER; REXFORD, 2008).

A Figura 2.4 (NICK FEAMSTER; REXFORD, 2008) mostra o conceito de locação de fatias de recursos físicos apresentados na arquitetura proposta no projeto CABO (CABO, 2005). Uma abordagem alternativa, utilizando "planos de rede" (*"Network Planes", ou NPs*) e "acordos de interconexão de redes" (*"Network Interconnection Agreements", ou NIAs*) pode ser encontrada no projeto AGAVE (BOUCADAIR et al., 2009).

2.3 Virtualização de Roteadores e Modelos de Consolidação

A virtualização de roteadores pode ser implementada de diferentes formas e ser aplicada em diferentes posições da rede. A combinação dos diversos arranjos possíveis forma os "modelos de consolidação" (KOLON, 2004) que são uma forma de classificar ou organizar os roteadores virtuais. A consolidação pode ser implementada nos dispositivos do núcleo (*core*) ou nos dispositivos da borda (*edge*) da rede e pode agrupar equipamentos de diferentes níveis da hierarquia de roteamento (consolidação vertical) ou equipamentos de um mesmo nível (consolidação horizontal).

Vale ressaltar aqui que existe uma certa diferença entre os termos roteador virtual (*virtual router*) e roteador lógico (LR, ou *logical router*). Para Matt Kolon (KOLON, 2004), *"Um roteador físico pode ser configurado com múltiplos roteadores lógicos baseados em software onde cada roteador lógico é uma partição dos recursos do roteador físico. Então cada roteador lógico baseado em software pode ser configurado com múltiplos roteamentos virtuais e instâncias de encaminhamento (VRFs ou Virtual Routing and Forwarding), onde cada VRF suporta uma VPN distinta.* ***Observe que um roteador virtual não é o mesmo que um roteador lógico.*** *Um roteador virtual é uma instância simplificada de roteamento que tem uma única tabela de roteamento. Um roteador lógico é uma partição de um roteador físico que pode conter múltiplas instancias de roteamento e tabelas de roteamento. Assim, um dado roteador lógico pode ser configurado para suportar múltiplas VPNs".* Neste trabalho, no entanto, estes termos são utilizados indistintamente como sinônimos, por questões de simplicidade e enfoque do estudo.

2.3.1 Consolidação Horizontal nas Bordas da Rede

A Figura 2.5 mostra a representação da consolidação horizontal nas bordas da rede. Neste modelo, múltiplos roteadores lógicos de borda são hospedados em um

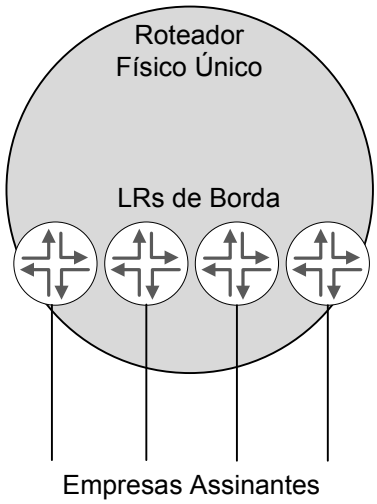


Figura 2.5: Consolidação Horizontal na Borda da Rede.

único roteador físico.

Um exemplo de aplicação deste modelo de consolidação pode ser utilizado em um grande provedor que tenha um grupo de operações do *backbone* e um outro grupo separado que gerencie os serviços para as empresas assinantes. Sem a virtualização, quando o grupo de serviços precisa instalar um novo roteador de borda no POP, ele deve comprar um roteador de borda, instalar uma interface 10 Gb, e então conectar o roteador de borda ao roteador de agregação/núcleo. Com a funcionalidade dos roteadores lógicos, o grupo de serviços pode configurar um roteador lógico com um número específico de interfaces de borda, configurar uma interface virtual de 10 Gbps para o roteador de agregação/núcleo, e gerenciar o roteador lógico utilizando as ferramentas de gerenciamento existentes.

Para haver eficiência de custos, a "consolidação horizontal na borda" requer que cada roteador lógico forneça uma granulosidade na configuração de suas interfaces virtuais, para que todos os roteadores lógicos possam compartilhar um enlace físico comum (*uplink*) ao roteador de agregação/núcleo do próximo salto. De outra forma, cada roteador lógico precisaria ter um enlace físico (*uplink*) dedicado e os benefícios econômicos de se instalar roteadores lógicos seriam reduzidos.

2.3.2 Consolidação Horizontal no Núcleo da Rede

A consolidação horizontal no núcleo da rede (Figura 2.6) combina múltiplos roteadores lógicos do núcleo, cada qual residindo no mesmo nível da hierarquia de roteamento, em um único roteador físico. Ainda que instalar múltiplos roteadores do núcleo em um único dispositivo físico possa impactar na confiabilidade da rede, a consolidação horizontal no núcleo é particularmente útil quando um provedor de serviços quer instalar, com baixo custo, redes paralelas sobre uma mesma infraestrutura física. O roteamento específico por aplicação (ASR, ou *Application Specific Routing* (KOLON, 2004)), pode ser implementado com o uso deste modelo.

2.3.3 Consolidação Vertical na Borda da Rede

Este modelo (Figura 2.7) é também chamado consolidação vertical na borda/agregação (ou agregação/núcleo) e combina múltiplos níveis da hierarquia de roteamento em um único roteador físico. Neste modelo, múltiplos roteadores lógicos

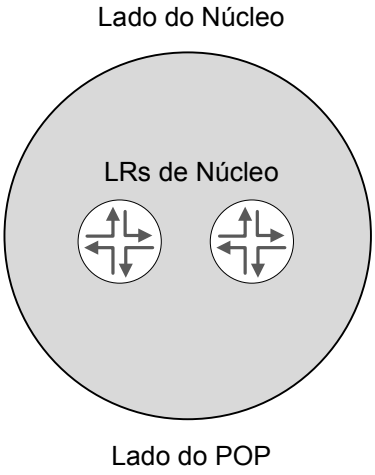


Figura 2.6: Consolidação Horizontal no Núcleo da Rede.

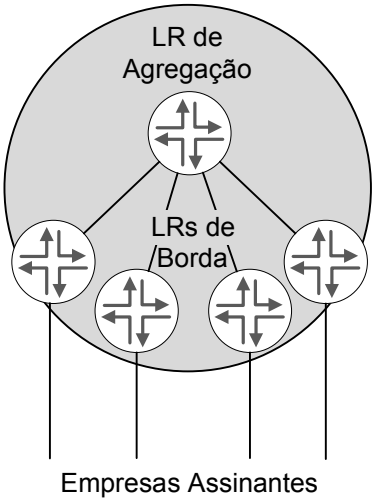


Figura 2.7: Consolidação Vertical na Borda da Rede.

de borda e um roteador lógico de agregação (ou múltiplos roteadores lógicos de agregação e um roteador lógico do núcleo) são configurados em uma única plataforma física de roteamento. A Figura 2.8 exemplifica este conceito.

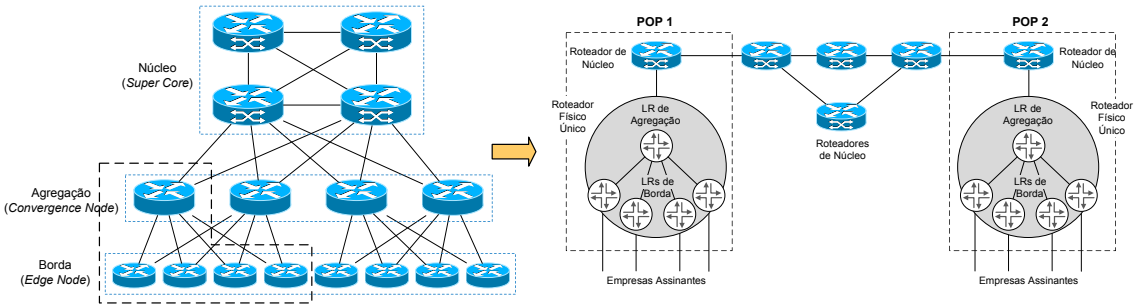


Figura 2.8: Exemplo de consolidação vertical na borda/agregação.

Na Figura 2.8, os quatro roteadores físicos de borda (*edge node*) e mais o roteador de agregação (*convergence node*) circundados pela linha pontilhada, à esquerda, foram agrupados em um único roteador com suporte à virtualização, representado

à direita, sob domínio administrativo do POP1. Estes dois níveis da hierarquia de roteamento são agora implementados por elementos virtuais (LRs) em um único equipamento físico.

Ainda que hospedar múltiplos níveis da hierarquia de roteamento em um único roteador físico possa potencialmente impactar na confiabilidade da rede, esta abordagem permite aos provedores manter sua hierarquia de roteamento de forma resumida, reduzir os custos de interconexão, e simplifica a topologia física da rede. O cenário de gerenciamento apresentado na seção 3.3.3 exemplifica o uso do modelo agregação/núcleo.

2.3.4 Consolidação Vertical no Núcleo da Rede

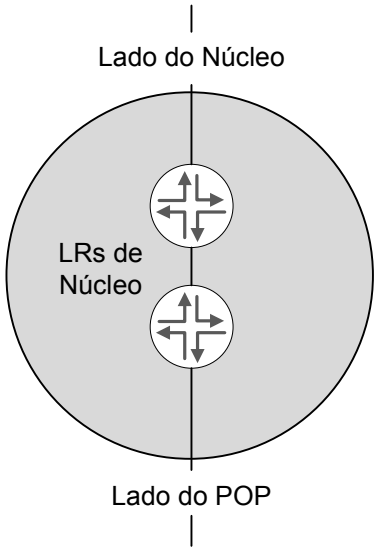


Figura 2.9: Consolidação Vertical no Núcleo da Rede.

A consolidação vertical do núcleo (Figura 2.9) combina múltiplos roteadores lógicos do núcleo, cada um dos quais situado em níveis diferentes na hierarquia de roteamento do núcleo, em um único roteador físico. Embora seja um modelo possível, este tipo de consolidação não oferece vantagens economicas ou operacionais conhecidas e por isso não é utilizado na prática. O cascateamento de dois roteadores em série, próximos um do outro, não traria benefício algum em relação a utilização de apenas um dispositivo deste tipo.

2.4 Trabalhos Relacionados

A literatura relacionada a virtualização de redes é extensa. Muitos trabalhos nessa área encontram-se relacionados às atividades de pesquisa sobre a Futura Internet (SPYROPOULOS; FDIDA; KIRKPATRICK, 2007). Neste contexto, a virtualização de roteadores é reiteradamente abordada. Alguns trabalhos particularmente importantes para esta dissertação são apresentados a seguir.

2.4.1 Trabalhos Acadêmicos

Um *survey* importante sobre a virtualização de redes foi apresentado por *Chowdhury e Boutaba* (CHOWDHURY; BOUTABA, 2009). Neste trabalho os autores

fazem uma revisão geral sobre o estado-da-arte nesta área, abordando os pricipais desafios e questões de pesquisa a serem estudados para realização de um ambiente virtual de redes (NVE, ou *"Network Virtualization Environment"*) e apresentando os principais projetos de pesquisa relacionados, em todo mundo. Vale destacar os desafios de pesquisa apontados com relação à Nodos e Enlaces Virtuais.

"A Virtual Internet Architecture" (TOUCH et al., 2003) é um trabalho pioneiro sobre a implementação de redes virtuais, onde são apresentados conceitos básicos os principais componentes destas redes e potenciais funcionalidades. Segundo os autores, uma "Internet Virtual" é uma rede IP composta de enlaces tunelados entre um conjunto de **roteadores virtuais** e *hosts* virtuais. Os conceitos desenvolvidos foram implementados e instalados na rede X-Bone (TOUCH, 2001). Apesar de focar a investigação sobre redes em *overlay*, são apresentados conceitos importantes sobre roteadores virtuais na seção II.2.2.

Um estudo técnico sobre aspectos da arquitetura de *hardware* para implementação de roteadores virtuais pode ser encontrado no trabalho conduzido por *Comer e Martynov* (COMER; MARTYNOV, 2006). A solução é baseada no uso de múltiplos processadores de rede (*"network processors"*) para suportar o roteamento virtual e em *overlays*.

O projeto AutoI (AUTOI, 2008) aborda o gerenciamento autonômico utilizando os principios de virtualização de redes para formar uma rede de recursos virtuais em *overlay*. O projeto é bastante atual e encontra-se em desenvolvimento, incluindo um extenso conjunto de publicações, documentações e códigos abertos para consulta.

2.4.2 Soluções Comerciais

Alguns equipamentos comerciais já possuem recursos para suportar a virtualização como os roteadores da família *Cisco XR 12000 Series* (CISCOXR, 2006) e os roteadores *Juniper TX Matrix Plus* (TXMATRIX, 2006). Estes recursos são acessados por meio dos sistemas operacionais destes dispositivos, que, embora providos de muitas funcionalidades, ainda não possuem interfaces suficientemente desenvolvidas para o gerenciamento completo e efetivo das configurações e operações associadas a virtualização.

A interface do IOS XR da Cisco atualmente fornece apenas dois comandos para gerenciar as funcionalidades de virtualização de seus dispositivos (CISCO, 2006): um para criar e outro para ligar um roteador lógico a um dispositivo físico. O IOS Junos da Juniper possuí uma interface (JUNOS, 2003) que possibilita a completa configuração de roteadores lógicos seguindo um procedimento bastante similar ao utilizado nos roteadores convencionais.

A Extreme Networks lançou em 2004 a série de roteadores BlackDiamond 10808 com suporte a virtualiazação na camada 3 (L3VS, 2007) (recurso posteriormente extendido aos demais equipamentos deste fabricante). Esses equipamentos disponibilizam uma interface por linha de comando (CLI) em seu IOS, o ExtremeXOS, para criar VRs e adicionar portas aos VRs, possibilitando entrar e sair na interface de configuração de cada VR a partir da CLI principal. A Extreme direcionou a virtualização principalmente para segmentação da rede em diferentes domínios administrativos. As portas de um switch de camada 3 só podem pertencer a um VR e os pacotes chegando na porta de um VR nunca podem ser comutados para as portas de outro VR. Dessa forma, estes dispositivos utilizam um conceito diferente do apresentado neste trabalho (Seção 2.1.1).

Todas estas interfaces comerciais disponíveis são basicamente constituídas por um sistema de configuração (CLI) que impacta indiretamente na sistemática de operação do roteador. Por isso, não abrange completamente o conjunto de funcionalidades potenciais do dispositivo, ou demanda um esforço considerável de parâmetros combinados com esta finalidade. Além disso, o conjunto de comandos varia de equipamento para equipamento, carecendo de padronização e interatividade (dinâmica) no acesso aos recursos.

2.4.3 Soluções Adaptadas

Diversos trabalhos encontrados na literatura buscam investigar a virtualização de redes por meio de soluções "adaptadas", que não foram nativamente desenvolvidas com este propósito. Como nesta dissertação, alguns autores utilizam plataformas de virtualização de *hosts* (Xen e/ou VMware, por exemplo) para emular o comportamento de roteadores virtuais e investigar aspectos como QoS, desempenho e a criação de avançados ambientes de teste (RODRíGUEZ-HARO; FREITAG; NAVARRO, 2009) (GALáN et al., 2009) (ANHALT; KOSLOVSKI; PRIMET, 2010). Outros preferem utilizar diretamente as técnicas de tunelamento (HOU; CHAN; YU, 2009) (TSUGAWA; FORTES, 2010), tipicamente implementadas em baixo nível pelas plataformas comerciais.

Todas estas abordagens visam estudar aspectos associados a virtualização de redes sem a utilização de dispositivos (nodos) originalmente projetados com este propósito. Embora estimativas de desempenho sejam relativas e limitadas a estas plataformas, estas soluções se beneficiam da flexibilidade nas implementações fornecendo ótimo conhecimento técnico sobre a operacionalidade destes sistemas.

2.4.4 VR-MIB

MIBlets (NG et al., 2002), são Bases de Informações de Gerenciamento (MIBs) particionadas, utilizadas para juntar e processar estatísticas de desempenho para cada uma das redes virtuais (VNs, ou *"Virtual Neworks"*) coexistentes em uma rede. Ao invés de usar uma MIB comum, cada nodo possui uma agente especial, com visão global de uma VN, gerenciando o estabelecimento, monitoramento e controle das conexões dos usuários para assegurar a correta alocação de recursos.

Algumas primitivas para o gerenciamento de roteadores virtuais foram desenvolvidas recentemente, como a migração de roteadores em tempo de execução (*live router migration*) (WANG et al., 2008) (AGRAWAL et al., 2006). O objetivo é tornar o procedimento de migração ou substituição de um roteador físico mais rápido e minimizar a "desrupção", utilizando chamadas remotas de métodos de uma API por um operador de rede.

Um módulo de MIB (*Management Information Base*) para roteadores virtuais foi definida em um *draft* do IETF (STELZER et al., 2003). Esse módulo de MIB foi desenvolvido no contexto de VPNs de nível 3, mas foi descontinuado. O módulo é composto de objetos relacionados a configuração em alto nível dos roteadores virtuais, bem como objetos para coleta de estatísticas sobre os dispositivos. A principal limitação deste módulo de MIB é que o mesmo não é adequado para operações como a migração de roteadores e a associação flexível de interfaces de rede reais e virtuais.

Apesar dos diversos trabalhos pesquisados, não foram encontradas soluções que implementem interfaces de gerenciamento adequadas para os roteadores virtuais

aplicados ao contexto da virtualização de redes da Futura Internet. Isso decorre, principalmente, do fato de não terem sido levantadas, em conjunto, as operações fundamentais que precisam ser realizadas para administrar o funcionamento destas redes.

3 SOLUÇÃO PROPOSTA

Em redes que utilizam virtualização, uma série de ações precisam ser executadas (manual ou automaticamente) e informações precisam ser administradas a fim de manter a operacionalidade dos sistemas. Algumas destas ações podem ser implementadas como primitivas de gerenciamento e virem a ser reutilizadas por outra(s) aplicação(ões) (*software* gerente). A utilização combinada de conjuntos de primitivas torna possível a criação de *frameworks* ou plataformas de gerenciamento capazes de realizar tarefas complexas de forma automatizada.

Neste capítulo são apresentadas as principais ações que podem ser implementadas como primitivas de gerenciamento (Seção 3.1). A seguir, é apresentado o projeto da interface de gerenciamento e a implementação do protótipo do agente desenvolvido neste trabalho (Seção 3.2). Por fim, são estudados alguns cenários de gerenciamento relacionados à solução proposta, a fim de analisar sua viabilidade em ambientes reais de rede (Seção 3.3).

3.1 Primitivas de Gerenciamento para Roteadores Virtuais

Com base na visão geral da virtualização de redes (Capítulo 2), esta seção foca nos conceitos específicos do gerenciamento dos nodos (roteadores virtuais), especialmente em aspectos de configuração dos dispositivos. Para isso, foi realizada uma análise do uso da virtualização de redes em ambientes documentados na literatura (Capítulo 3.3) e também um levantamento dos potenciais recursos necessários a estes sistemas.

3.1.1 Criação de um Roteador Virtual

A operação de criação de um roteador virtual deve ser realizada sempre que for necessário alocar um novo roteador em uma determinada localidade. Conforme explicado na Seção 2.3, isto pode ser feito na borda ou no núcleo da rede.

A criação de um roteador virtual pode ser feita de diferentes formas e em diferentes momentos. Quando o equipamento (substrato ou plataforma física) está realizando o roteamento de pacotes, em particular, pode ocorrer interferência da operação de criação de um novo VR sobre o tráfego dos usuários por conta da demanda por processamento, ocasionando alguma redução na vazão de pacotes. Além disso, as diferentes cargas computacionais do *hardware*, ocasionadas pelos protocolos de roteamento, podem influenciar no desempenho da operação de criação de VRs. Esta operação pode ser classificada nos seguintes tipos:

i. Criação de um novo roteador virtual: neste caso o novo roteador virtual é

criado sem um conjunto pré-definido de configurações para sua operação;

ii. Criação a partir de um *template*: neste caso o roteador virtual é criado com um conjunto de configurações anteriormente definido para instalá-lo de forma mais ágil.

Embora visível ao usuário final, esta separação não apresenta diferenças significativas de implementação. Na prática, a criação de um novo roteador virtual sempre é feita a partir de um *template*, porém no caso i) o modelo já vem pré-definido do fabricante. No caso ii), existe, primeiramente, o processo de configuração de um VR e a geração de uma imagem para ser utilizada como modelo. Esta customização pode ou não ser útil, dependendo da escala e diversidade das plataformas nas quais os VRs serão instalados.

3.1.2 Remoção de um Roteador Virtual

A remoção de um roteador virtual consiste em apagar ou inutilizar o mesmo. Diferentemente dos equipamentos reais, esta operação não implica em prejuízos materiais, o que demonstra a flexibilidade e economicidade da virtualização. Quando um VR é removido, ele pode *i*) ser apenas desregistrado do sistema, ficando sua imagem ou configuração armazenada; ou *ii*) ser completamente apagado.

Remoções são necessárias quando há alguma mudança de demanda (como, por exemplo, nos casos em que se faz reserva automática de banda para tráfego - *"bandwidth on demand"* (URUSHIDANI et al., 2008) - criando e removendo roteadores virtuais conforme as rajadas de tráfego se apresentam) ou quando um equipamento atinge seu limite de capacidade de alocação de recursos.

3.1.3 Inicialização de um Roteador Virtual (Ligar)

O comando de inicialização de um roteador virtual corresponde a operação de ligar um roteador real (que pode ocorrer energizando diretamente o dispositivo ou por meio de uma chave *on/off*). O conceito é o mesmo: ao ser ligado, um VR irá inicializar seu sistema operacional carregando o conjunto de configurações previamente salvas. Logo, a inicialização é sempre sucedida de um intervalo de tempo (*boot time*) variável que o equipamento leva para inicializar o VR. Esta primitiva pode ser facilmente associada a criação de um VR (seção 3.1.1) para formar uma operação integrada do tipo cria-e-iniciliza (*"create-and-go"*).

3.1.4 Interrupção de um Roteador Virtual (Desligar)

O comando de interrupção de um roteador virtual corresponde a operação de desligar um roteador real. A interrupção pode ocorrer de duas formas:

i. Parada convencional: em que o comando disparado aguarda a finalização do sistema operacional do VR (similiar ao desligamento de um computador pessoal);

ii. Interrupção abrupta: neste caso o roteador virtual é interrompido imediatamente, sem aguardar a finalização do sistema operacional (similar a tirar da tomada um computador pessoal).

3.1.5 Migração de um Roteador Virtual

A operação de migração consiste em transferir um roteador virtual de um equipamento físico (substrato) para outro. Esta operação pode ser bastante útil nos casos de manutenção de equipamentos, bem como na realocação de recursos. A migração elucida o alto grau de desacoplamento *hardware/software* presente na virtualização e pode ser realizada, basicamente, de duas formas:

i. Em tempo de execução (*"live migration"*): em que o VR é migrado de forma rápida, sem apresentar qualquer tempo de indisponibilidade dos serviços;

ii. Desativação temporária: neste caso, o roteador virtual é interrompido por alguns instantes até ser novamente ativado em um novo equipamento. O tempo de indisponibilidade é variável e existem alguns trabalhos na literatura (AGRAWAL et al., 2006) que visam otimizar este processo.

Embora seja uma operação complexa, que possa ser implementada por um conjunto de primitivas, já existem trabalhos que implementam a migração como uma única primitiva (WANG et al., 2008). Soluções similares já foram comercialmente implementadas em servidores (XEN, 2003). A migração de um roteador geralmente deve ser sucedida pela migração dos enlaces, a fim de direcionar os tráfegos adequadamente.

3.1.6 Migração de um Enlace

No nível IP, os equipamentos possuem um endereço IP associado a uma interface de rede pela qual os pacotes entram (*downlink*) e saem (*uplink*) do dispositivo. O direcionamento dos pacotes é tipicamente realizado por uma tabela de roteamento. A migração de enlace consiste em alterar a rota ou direcionamento dos pacotes de um enlace físico para outro. Esta operação é extremamente importante nos casos de migração, em que um VR muda de posição na rede e os roteadores conectados a ele precisam apontar o novo caminho (interface de rede) para seu tráfego alcançá-lo. Com isso, os roteadores vizinhos a um VR migram seus enlaces para atingir o VR.

Nos *backbones* de transporte, os roteadores são normalmente interconectados por meio de um caminho óptico através de *switches* programáveis (redes ASON). Nessas redes, a mudança do caminho óptico pode ocorrer em escalas de sub-nanosegundos (ROSTAMI; SARGENT, 2006).

A migração pode ocorrer em dois tipos de enlaces:

i. Enlaces de Acesso: enlaces que conectam roteadores de borda do cliente (CE, ou *Customer Edge*) a roteadores de borda do provedor (PE, ou Provider Edge);

ii. Enlaces de Núcleo: enlaces que interconectam dois dispositivos de dentro da nuvem do provedor de serviços (disposivos P, de *Provider*). Estes dispositivos não se conectam diretamente às redes dos clientes e desconhecem VPNs ou túneis estabelecidos por estes.

A migração de enlaces também é bastante importante para a tolerância a falhas. Nestes casos, a falha de um enlace deve ser rapidamente detectada e o tráfego chaveado (redirecionado) para uma nova interface de rede. Tecnologias como o FRR (*Fast Route Recovery*) do MPLS implementam esta funcionalidade.

3.1.7 Criação de uma Interface Virtual

Uma interface virtual de rede é o elemento lógico correspondente a uma interface de rede (NIC) de um roteador real. As interfaces virtuais precisam ser vinculadas (*"binded"*) a interfaces reais do *hardware*. Dessa forma, a operação de criação de uma interface virtual deve informar, entre outros parâmetros, à qual enlace físico do dispositivo a mesma estará associada.

Roteadores virtuais "enxergam" interfaces virtuais como se fossem interfaces físicas. A plataforma de virtualização deve gerenciar tanto o vínculo de interfaces virtuais com intefaces físicas quanto a visibilidade destas intefaces a um determinado roteador virtual, que só irá ter acesso às interfaces registradas a ele. A virtualização das interfaces de rede sempre traz algum *overhead* de desempenho que depende da implementação e do nível de compartilhamento demandado sobre uma interface física.

Da mesma forma que ocorre com os VRs, a criação de VIs (*Virtual Interfaces*) sempre ocorre a partir de um ou mais *templates* disponíveis. Estes modelos são previamente configurados (em tempo de instalação) e contém informações como o tipo de interface (ex., interface Ethernet 100 Mb/s) e a forma como é feito o mapeamento lógico/físico (ex., por tradução de endereços NAT).

3.1.8 Remoção de uma Interface Virtual

A remoção de uma interface virtual consiste em desregistrar a mesma do sistema. Diferentemente do que ocorre com os VRs, os modelos (ou *templates*) das interfaces virtuais não podem ser diretamente excluídos, já que são configurados durante a instalação da plataforma de virtualização. Da mesma forma que na criação de VIs, na remoção deve ser informado a qual VR a VI pertence, já que as VIs estão associadas a VRs.

3.1.9 Ativação de uma Interface Virtual

A ativação de uma interface virtual é utilizada para fazer o *"hot plugging"* do componente. Quando um VR está sendo executado, a simples criação de uma interface virtual, não faz com que a mesma esteja disponível (ou visível) para o VR. Isto só é possível através do comando de ativação da VI, ou re-inicializando o VR para que ele detecte a nova interface. Dessa forma, a ativação de uma VI traz ainda mais flexibilidade para configuração e operação dos componentes virtualizados.

3.1.10 Desativação de uma Interface Virtual

A desativação de uma interface virtual é uma operação menos comum que corresponde a indisponibilizar a interface para uso do VR. Esta operação é bastante similar a mudança de estado do tipo *down* que pode ser configurada de forma manual diretamente no sistema operacional do VR. Além disso, a desativação de uma interface poderia ser realizada simplesmente removendo diretamente a mesma - mesmo que ligada - o que também torna o uso desta primitiva menos frequente.

3.1.11 Outras Operações Sobre Roteadores Virtuais

As subseções anteriores apresentaram as operações que poderiam ser implementadas como primitivas de gerenciamento. Muitas outras operações são importantes para o funcionamento dos ambientes de redes virtuais, como a leitura de informações

de estado dos equipamentos ou a configuração individual dos roteadores virtuais. Estas ações, no entanto, não caracterizam operações específicas para virtualização de redes, mas sim atividades inerentes a diversos tipos de redes. A virtualização dos elementos de rede não exclui a possibilidade de gerenciar alguns de seus parâmetros e comportamentos, comuns aos componentes reais, de forma similar ou até mesmo idêntica. As primitivas apresentadas neste capítulo (com exceção da migração de enlaces), embora análogas a procedimentos reais, no entanto, são exclusivas dos roteadores e equipamentos que utilizam a virtualização.

Este trabalho está focado na definição de objetos e operações para o gerenciamento de roteadores virtuais. Algumas das operações utilizadas no gerenciamento de *hosts* virtuais e roteadores convencionais são as mesmas que aquelas utilizadas no gerenciamento de roteadores virtuais, podendo gerar alguma confusão com relação a distinção entre o gerenciamento destes dipositivos e o de roteadores virtuais. O gerenciamento de roteadores virtuais inclui a implementação de operações de rede que são particulares para virtualização de redes. Conforme apresentado nesta seção, estas operações podem ser implementadas com primitivas de gerenciamento (ou combinações delas). Algumas destas primitivas, como a criação e remoção de roteadores virtuais, são praticamente as mesmas daquelas encontradas em máquinas virtuais, mas outras, como a criação e remoção de interfaces virtuais, ou a migração de roteadores e enlaces virtuais, são bastante distintas, tanto em existência como na sua implementação. É natural que exista alguma intersecção do gerenciamento de roteadores virtuais com ambos o gerenciamento de máquinas virtuais (algumas operações e mecanismos de endereçamento em comum, por exemplo), já que ambos utilizam os principios da virtualização, e com o gerenciamento de roteadores tradicionais (como a configuração e coleta de estatísticas dos dispositivos), já que ambos são plataformas de gerenciamento. Além disso, o gerenciamento de ambientes de redes virtuais envolve uma série de outras questões de pesquisa importantes para formação de uma plataforma completa de gerenciamento (CHOWDHURY; BOUTABA, 2009), como o particionamento de recursos e atribuições entre InPs e SPs, os quais não são foco desta dissertação. Assim, as disciplinas de gerenciamento de redes tradicionais, gerenciamento de *hosts* virtuais (servidores, recursos em núvem, etc.) e gerenciamento de redes virtuais não devem ser confundidas.

3.2 Projeto e Implementação do Protótipo

A tecnologia SNMP é amplamente utilizada para o gerenciamento de redes. Este protocolo é uma alternativa natural de escolha para lidar com roteadores, quando considerados requisitos como operacionalidade, interoperabilidade e reuso. As primitivas de gerenciamento estudadas neste trabalho foram implementadas com este protocolo a fim de avaliar seu funcionamento e desempenho. Desta forma, nessa seção, é apresentado o projeto e implementação do protótipo de um agente SNMP desenvolvido para as plataformas VMware e Xen.

Embora o SNMP seja raramente utilizado para configuração (SCHONWALDER; PRAS; MARTIN-FLATIN, 2003), ignorá-lo como uma possível interface para virtualização não seria apropriado. As primitivas estudadas são operações básicas que, por sua natureza, possuem poucos objetos necessários à sua implementação, o que não traria benefícios significativos com o uso de soluções baseadas em Web Services (FERRIS et al., 2002), por exemplo. Além disso, a utilização direta de protocolos

mais complexos, como o NETCONF (ENNS, 2006) não agregaria benefícios imediatos ao trabalho, ainda que, presumivelmente, possa ser uma escolha adequada no desenvolvimento de solução mais avançadas que combinem conjuntos de primitivas. Dessa forma, o protocolo SNMP foi escolhido como um ponto inicial para identificação dos requisitos gerais para interfaces de gerenciamento aplicadas à virtualização de redes.

3.2.1 Modelo de Referência

A grande flexibilidade de desacoplar *hardware* e *software* trazida pelas tecnologias de virtualização, trouxe consigo o aumento de complexidade que pode ser criado quando diversas redes de larga escala são sobrepostas. A necessidade de uma alto grau de automação e inteligência nestes ambientes pode ser primeiramente viabilizado utilizando-se agentes SNMP com as informações de MIB apropriadas que podem ser utilizadas por *frameworks* de gerenciamento mais avançados.

A configuração de alguns recursos virtualizados pode incluir o dispositivo inteiro ou apenas algumas partes de seu *hardware*, como as interface de rede. Dessa forma, um único roteador físico com n ($n \geq 1$) interfaces reais de rede pode emular um ou mais roteadores virtuais com m ($m \geq 1$) interfaces de rede virtuais cada.

Um exemplo de arquitetura para roteadores virtuais foi apresentado na proposta VROOM (WANG et al., 2008). Este trabalho apresenta uma extensão para esse modelo a fim de contemplar uma interface de gerenciamento para o dispositivo.

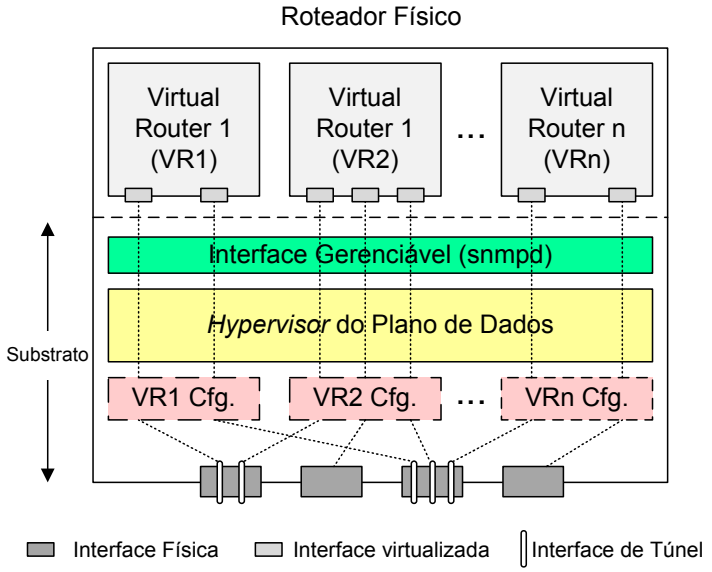


Figura 3.1: Arquitetura de roteador virtual gerenciável.

A Figura 3.1 mostra a arquitetura de nodo proposta a fim de contemplar uma interface de gerenciamento para roteadores virtuais. O roteador físico foi dividido em duas camadas: uma camada de baixo nível, ou substrato, onde uma visão global do dispositivo está disponível e as funcionalidades do plano de dados são executadas, e uma camada de alto nível, em que roteadores virtuais isolados implementam o plano de controle. A solução de gerenciamento proposta também está dividia em dois planos: no nível mais baixo, funções globais do sistema (como a criação, remoção e migração de roteadores virtuais ou mapeamentos de interfaces de rede) são implementadas por um agente snmp (snmpd), uma fina camada de *software* sendo

executada no topo da aplicação que é utilizada para controlar o *hardware* (*hypervisor*), enquanto no nível mais alto as atividades do plano de controle (ex., protocolos de roteamento) são realizadas. Nesta abordagem, cada roteador virtual contém 2 (dois) níveis de configuração: um nível de configuração baixo (VR1 Cfg.,VR2 Cfg., etc) que controla as associações de interfaces de rede e um nível de configurações alto (interno a cada roteador virtual) que controla os roteamentos e outras funcionalidades relacionadas ao plano de controle. Dessa forma, existem duas entidades que precisam ser gerenciadas: o substrato e os diferentes roteadores virtuais hospeados sobre ele. Em geral, existem duas abordagens para se fazer isso:

i) **Um agente central para gerenciar todo sistema:** nesta abordagem, um único agente SNMP colocado no substrato (como mostrado na Figura 3.1) pode gerenciar ambos o roteador hospedeiro e os roteadores virtuais hospedados sobre ele. A comunicação de gerenciamnto da rede é estabelecida entre o gerente e este agente global, que pode então acessar ambas MIBs da camada inferior e superior. O endereçamento é feito utilizando contextos SNMP (STELZER et al., 2003) (HARRINGTON; PRESUHN; WIJNEN, 2002) e pode utilizar *strings* de comunidade SNMPv2c ou o *contextName* (nome de contexto) SNMPv3. Por exemplo, mensagens *set* ou *request* endereçadas para um dispositivo possuindo 2 (dois) VRs (ex., "vr01" e "vr02") seriam interpretadas pelo agente SNMP central, que poderia acessar tanto "vr01" ou "vr02", como também o substrato (ex., "admin"). Para trocar informações com os roteadores virtuais, o agente SNMP pode usar as seguintes técnicas: enviar comandos diretamente para o sistema operacional de cada VR, por meio da API do *hypervisor* (se este suportar isso); ou encaminhar as mensagens, atuando como um *gateway*, para algum serviço de escuta presente internamente em cada VR. Este serviço poderia ser implementado por um *daemon* SSH, RPC, ou mesmo SNMP. Uma última opção seria utilizar o protocolo AgentX para implementar agentes SNMP extensíveis (DANIELE et al., 2000).

ii) **Múltiplos agentes:** neste caso, cada entidade gerenciada contém um agente SNMP completo que pode acessar sua própria MIB. A comunicação de gerenciamento de rede pode ser estabelecida entre o gerente e o agente de cada roteador virtual. Com isso, o balanceamento inteligente da carga de processamento entre *multi-cores* pode ser obtido. O endereçamento é alcançado através de mecanismos UDP/IP comuns, já que os dispoitivos físicos e virtuais contém seus próprios endereços. As mensagens são diretamente recebidas e processadas por cada agente.

Em qualquer caso, quando o agente SNMP (*"snmpd"*) recebe uma mensagem de requisição do gerente, ele emite um comando para o *hypervisor* do sistema a fim de recuperar ou acionar variáveis SNMP. Uma API (*Application Programming Interface*) deve estar disponível no lado do *hypervisor* com a finalidade de habilitar as operações do agente SNMP. Como alternativa, no caso de indisponibilidade de uma API, o dispositivo físico poderia também ser gerenciado editando alguns de seus arquivos de configuração, mas isso iria aumentar significativamente a complexidade de implementação do agente. Este capítulo é dedicado aos problemas do núcleo de gerenciamento e foca no agente SNMP e sua MIB associada para o gerenciamento da virtualização através da interação com o *hypervisor* do roteador físico.

3.2.2 MIB Virtual Router Extendida

Basicamente, o gerenciamento de roteadores virtuais requer informações que podem ser classificadas em *i*) configurações do dispositivo; *ii*) associações de interfaces de rede; e *iii*) estatísticas do dispositivo. Neste trabalho, a MIB *Virtual Router* (STELZER et al., 2003), originalmente definida em uma versão rascunho (*Internet draft*) foi utilizada como base para a interface de gerenciamento proposta. Embora seja uma versão antiga e expirada de um *draft*, esta abordagem foi preferida em relação a escrever uma MIB completamente nova, já que as informações encontradas no *draft* são pertinentes a situação atualmente estudada.

A MIB *Virtual Router* original cobre algumas operações fundamentais como criação e remoção de roteadores virtuais, assim como a associação entre uma interface de rede virtual e uma física. O conjunto original de objetos da MIB, no entanto, não é suficiente para lidar com todas operações necessárias para o gerenciamento básico de uma rede virtual. Por exemplo, a MIB *Virtual Router* não suporta associar mais do que uma interface virtual a uma única interface física ao mesmo tempo. A fim de melhorar a MIB *Virtual Router* é proposta uma extensão a esta MIB apresentada na Figura 3.2. A especificação desta MIB em SMI (*"Structure of Management Information"*) encontra-se no Apêndice A.1.

Na Figura 3.2, os itens destacados entre o pontilhado correspodem aos objetos acrescentados à MIB e as siglas NA, RO e RC correspondem, respectivamente, aos objetos não acessíveis (*not-accessible*, utilizados como indexadores auxiliares), acessíveis somente em leitura (*read-only*) e acessíveis em leitura, escrita e criação (*read-create*) (MCCLOGHRIE; PERKINS; SCHOENWAELDER, 1999).

O grupo **`vrConfig`** contém informações sobre a indexação, criação e remoção de VRs. A variável `vrId` (*virtual router identifier*) é um identificador único iniciando em 1 (vrId 0 é reservado para o "VR Administrativo") que enumera cada roteador virtual instalado. Para auxiliar na atribuição de um "vrId" sem conflito, a estação de gerenciamento pode recuperar o próximo identificador disponível de um VR acessando o escalar `vrConfigNextAvailableVrId` do dispositivo gerenciado. Cada configuração de um roteador virtual é armazenada na `vrConfigTable` que contém informações como o identificador do roteador virtual (`vrId`), nome (`vrName`) e estado (`vrRowStatus`). A coluna de estado é utilizada para criar e remover roteadores virtuais e implementa 3 operações de acordo com os valores definidos: *i*) "inicializar" ou "parar" um VR, através dos valores ACTIVE ou NOTINSERVICE, respectivamente; *ii*) "criar"ou "criar e inicializar"um VR, através dos valores CREATEANDWAIT ou CREATEANDGO, respectivamente; *iii*) "remover" um VR, através do valor DESTROY. Caso o VR esteja sendo utilizado e um DESTROY seja solicitado o VR sofrerá uma parada forçada e será então removido. Além disso, a coluna de estado pode conter um valor neutro (*idle*) que serve para indicar que o dispositivo não está pronto ou possui algum erro interno (valor NOTREADY). As outras informações desta tabela se referem a configurações internas de cada dispositivo virtual e incluem o identificador de VPN (`vrVpnId` ou *Virtual Private Network Identifier*), o máximo número de rotas suportadas (`vrMaxRoutes`) e um gatilho para incializar ou desligar protocolos de roteamento (`vrRpTrigger`).

O grupo **`vrStat`** contém objetos escalares com informações globais do dispositivo (ex., `vrConfiguredVRs` - o número de VRs configurados em cada nodo; e `vrActiveVRs` - o número de VRs que estão ativos nestes nodos) e uma tabela com o estado administrativo e operacional de cada VR (`vrStatTable`). Esta tabela é

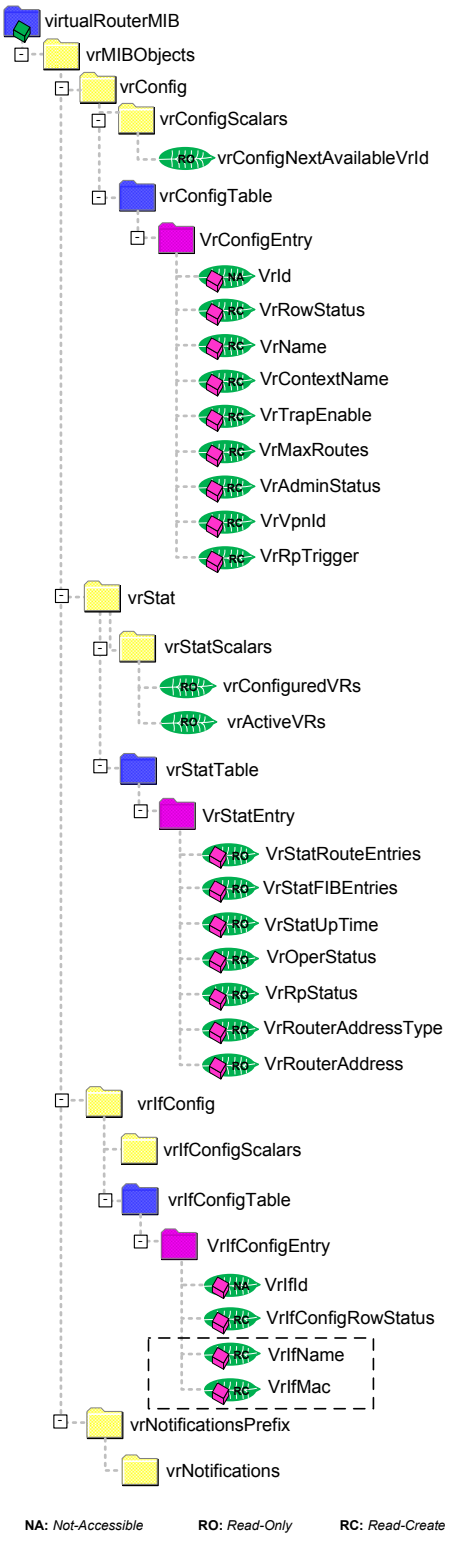


Figura 3.2: MIB *Virtual Router* Extendida.

indexada pela variável `vrId` e armazena estatísticas como o número atual de entradas de rota (`vrStatRouteEntries`), o endereço IP da interface de rede principal de um VR (`vrRouterAddress`) e o tempo decorrido desde que o VR começou a operar (`vrStatUpTime`).

O grupo **`vrIfConfig`** possui uma tabela para configuração das interfaces de

rede virtuais (`vrIfConfigTable`). Esta tabela armazena todas associações entre os roteadores virtuais e as interfaces físicas de rede. A tabela é indexada por ambos `vrId` e `vrIfId`. O `vrIfId` é o identificador único para cada interface virtual de um mesmo roteador virtual. Como na tabela `vrConfigTable`, as interfaces de um roteador virtual podem ser criadas, removidas, ou ter seu estado operacional modificado através da edição da variável `vrIfConfigRowStatus`. Esta coluna pode ter os valores (1) `active`, (2) `notInService`, (3) `notReady`, (4) `createAndGo`, (5) `createAndWait`, ou (6) `destroy` e é utilizada para disparar as operações de criação e remoção ou para determinar o estado operacional das interfaces virtuais de cada VR. Para criar uma nova interace de um VR, o agente SNMP primeiramente verifica se o `vrId` correspondente existe e se o `vrIfId` escolhido está disponível naquele VR.

O grupo **`vrNotifications`** permite ao gerente habilitar ou desabilitar *traps* manipulando a variável `vrTrapEnable` da tabela `vrConfigTable`. Estas *traps* incluem `vrUp`, `vrDown`, e `vrMaxRoutesExceeded`.

3.2.3 Operações de Exemplo

Nesta seção é exemplificado como as variáveis para a MIB proposta são manipuladas a fim de executar algumas operações para o gerenciamento de roteadores virutais.

Criação de Roteadores Virtuais

Para criação de roteadores virtuais, as variáveis `vrRowStatus` e `vrName` são utilizadas da seguinte maneira:

- Recuperação do próximo ID disponível para o Roteador Virtual utilizando o `vrConfigNextAvailableVrId`:

```
GetRequest {vrConfigNextAvailableVrId.0}
Response   {vrConfigNextAvailableVrId.0 = 9}
```

- Na `vrConfigTable`, criar uma instância de um VR utilizando o `vrRowStatus`:

```
SetRequest {
   vrRowStatus.9  createAndGo(4),
   vrName.9  "TestVR"
}
```

Outros parâmetros também podem ser utilizados durante a criação de um novo VR, como o nome do contexto (`vrContextName`), o estado administrativo (`vrAdminStatus`) e o *flag* para habilitar *traps* (`vrTrapEnable`). Em caso de sucesso, o novo roteador virtual é criado e inicializado e a variável `vrConfigNextAvailableVrId` é incrementada por 1.

Remoção de Roteadores Virtuais

Para remoção de um roteador virtual, é utilizada a variável `vrRowStatus` apenas:

```
SetRequest {vrRowStatus.9  destroy(6)}
```

Esta ação involve parar o roteador virtual antes de remove-lo. Isto pode levar a um atraso extendido (parada convencioanal) ou não (parada forçada), dependendo de como a parada é realizada.

Associação Dinâmica das Interfaces

Gerenciar interfaces de rede virtuais é outro aspecto fundamental que deve ser tratado a fim de construir uma solução de gerenciamento para virtualização de redes. Assim como para os roteadores virtuais, as interfaces de rede virtuais podem ser criadas, configuradas e removidas. Mais especificamente, as interfaces de rede virtuais precisam ser associadas a um interface de rede física. Essa vinculação é chamada associação (*binding*) ou mapeamento de interfaces. Para suportar essas funcionalidades, duas variáveis foram adicionadas à `vrConfigTable`:

- A variável **`vrIfName`** armazena o nome da interface física de rede (NIC) (ex., "eth0", "eth1"), sendo responsável pela associação de uma maneira geral entre a interface física e lógica. O nome é "setado" durante a criação da interface e pode ser re-configurado mais tarde;
- A variável **`vrIfMac`** armazena o endereço MAC da interface virtual. O usuário pode especifica-lo manualmente ou utilizar o valor especial `generate`; neste caso, o agente automaticamente alocará um novo endereço MAC não utilizado. Se não for especificado durante a criação, o `vrIfName` assumirá o valor `generate` por padrão.

Um exemplo de criação de uma instância de interface para um VR utilizando o `vrIfConfigRowStatus` da `vrIfConfigTable` é apresentado a seguir:

```
SetRequest {
    vrIfConfigRowStatus.1.3  createAndGo(4),
               vrIfName.1.3  "NAT_vmnet8",
                vrIfMac.1.3  "generate"
}
```

Neste caso o agente irá acionar a criação de uma nova interface com o índice 3 para o roteador virtual 1 e ativa-la (*up*); esta operação pode ser realizada com sucesso mesmo quando o roteador virtual está sendo executado (*hot plugging*). Se o roteador virtual especificado não existe ou outra interface identificada com o índice 3 já existir, a operação irá falhar. Ambos os parâmetros `vrIfConfigRowStatus` e `vrIfName` são obrigatórios durante a criação da interface do VR, de forma que se algum deles não estiver presente na varbind (*varbind list*), a operação deverá falhar. A configuração dinâmica das associações pode ser obtida modificando o nome da interface do roteador virtual, por exemplo:

```
SetRequest {vrIfName.1.3  "Bridged_eth0"}
```

Por fim, a remoção de uma interface pode ser realizada setando o estado da linha (*row status*):

```
SetRequest {vrIfConfigRowStatus.1.3 destroy(6)}
```

3.2.4 Implementação do agente SNMP

O código para o novo módulo de MIB foi implementado (Apêndice A.2) e anexado a um agente SNMP que trata as requisições e executa as operações definidas na seção 3.1. A implementação proposta utilizou como base o pacote NET-SNMP 5.5 (NET-SNMP, 2000) e o comando "mib2c" para gerar o esqueleto do código desenvolvido (Apêndice A.3) e um *script* para recompilá-lo após quaisquer modificações (Apêndice A.4). O agente foi integrado através de uma plataforma de linha de comando (CLI, ou *Command Line Interface*) por *scripts bash* com as funções da API subjacente oferecidas pelo *hypervisor* das plataformas de virtualização VMware Server (VMWARE, 1998) e Citrix Xen Server (XEN, 2003), nas quais o agente foi instalado e avaliado. O Xen usa o comando "xe" associado a um conjunto de parâmetros para gerenciar seus recursos virtuais (ex., máquinas virtuais, discos e interfaces de rede), enquanto que o VMware utiliza ambos os comandos "vmrun" e "vmware-vim-cmd".

As ações para criar elementos virtuais são disparadas por mensagens SNMP `set request`. Um exemplo é a criação de um novo roteador virtual:

```
snmpset -v 2c -c private localhost vrId.1 u 1 vrRowStatus.1
i 4 vrName.1 s "VRouter1" vrContextName.1 s "vr0"
vrTrapEnable.1 i 1 vrAdminStatus.1 i 1
```

Neste caso foi utilizada a "versão 2c" do protocolo SNMP (-v 2c) para enviar uma requisição do tipo "setrequest" (snmpset), de um usuário utilizando a "comunidade private" (-c private) "endereçada ao agente SNMP localizado no próprio host" de onde partiu a requisição (localhost) passando a varbind list com valores para os objetos `vrId`, `vrRowStatus`, `vrName`, `vrContextName`, `vrTrapEnable` e `vrAdminStatus` (ver Figura 3.2, para referência, e Apêndice A.1, para descrição dos objetos) das entradas da primeira linha (.1) da tabela correspondente (`vrConfigTable`, neste caso).

A resposta obtida em caso de sucesso na execução da linha de comando supra citada, indicando o tipo e valor de cada parametro configurado, seria:

```
VIRTUAL-ROUTER-MIB::vrRowStatus.0 = INTEGER: createAndGo(4)
VIRTUAL-ROUTER-MIB::vrName.0 = STRING: VRouter1
VIRTUAL-ROUTER-MIB::vrContextName.0 = STRING: vr0
VIRTUAL-ROUTER-MIB::vrTrapEnable.0 = INTEGER: true(1)
VIRTUAL-ROUTER-MIB::vrAdminStatus.0 = INTEGER: up(1)
```

Mensagens SNMP, no lado do agente, resultam na execução de *scrips bash* (Apêndice A.5) que contactam o *hypervisor* e o sistema operacional ou editam algum arquivo de configuração. Um típico comando para criação de um roteador virtual no Xen iria parecer com o seguinte:

```
xe vm-install template=(vyatta-vm-template) new-name-label=
(name-of-the-vm)
```

No VMware, a mesma operação pode ser realizada por cópia e colagem (*copy and paste*) da imagem da máquina virtual e então registra-la com:

```
vmrun -T server -h https://localhost:8333/sdk -u $HOST_USER
-p $HOST_PASS register ''[$DATASTORE_NAME] $TEMPLATE_IMG.vmx''
```

A associação das interfaces de rede foi efetuada utilizando algumas funções das APIs e/ou editando diretamente os arquivos de configuração das máquinas virtuais. Editar diretamente arquivos de configuração não é uma solução apropriada já que aumenta a complexidade e pode levar a comportamentos inesperados, mas foi utilizado devido a limitações da API do *hypervisor* disponível no VMware Server. No VMware, o objeto `vrIfName` corresponde a combinação entre o tipo de rede (*bridged, host only* ou *nat*) e a interface física (*eth0, eth1, ..., ethN*), enquanto no Xen este objeto contém o identificador único da interface física (*network-uuid*). Quando requisitado a editar ou criar um nova placa ou interface de rede virtual, o agente SNMP verifica se os parâmetros recebidos da *varbind list* são suficientes e válidos.

Algumas questões importantes de projeto devem ser observadas no desenvolvimento do agente SNMP.

a) O agente deve retomar o estado atual do dispositivo. Para as informações dos VRs e VIs serem consultadas (snmpget) ou escritas (snmpset), estes dados devem ter sido cadastrados previamente junto à base de informações interna do agente, para que este possa fazer a verificação de consistência e viabilidade das ações demandadas. Uma opção é cadastrar estas informações no momento de inicialização do agente e ir atualizando o dispositivo por meio de comandos disparados diretamente ao agente SNMP. Neste caso, outras interfaces de operação (ex., CLI) devem ser desabilitadas ou não utilizadas (a menos que disparem eventos de atualização ao agente) para evitar inconsistências. Outra alternativa seria atualizar as informações de estado do dispositivo a cada acesso ao agente, mas nesse caso deve ser considerado o custo de acesso às APIs subjacentes para recuperar estes dados, bem como questões de consistência em acessos simultâneos;

b) Deve ser verificada a consistência sintática e semântica dos parâmetros acionados. Em particular, nos acessos em escrita, o agente deve verificar se "pode" ou "não pode" realizar a operação solicitada. Por exemplo, a criação de um novo roteador virtual com um `vrId` que já exista, não pode ser realizada, pois sobrescreveria o elemento existente. A criação de interfaces virtuais só pode ser realizada se o `vrId` for válido (existente) e o `vrIfId` estiver disponível. Valores inválidos para objetos como o `vrRowStatus` e `vrIfConfigRowStatus` não podem ser aceitos por não terem qualquer ação associada;

c) Informações internas podem ser necessárias. O agente pode precisar armazenar informações em suas estruturas de dados internas que não possuam objetos diretamente acessíveis para garantir a transparência da interface ao usuário. Este tipo de adaptação é mais comum nos casos em que a API subjacente disponibilizada pelo hypervisor não possui recursos suficientes para realizar todas ações necessárias. O agente, pode, por exemplo, precisar relacionar algum identificador externo do roteador virtual associado ao vrId interno ou ainda precisar armazenar o caminho para algum arquivo de configuração do roteador virtual;

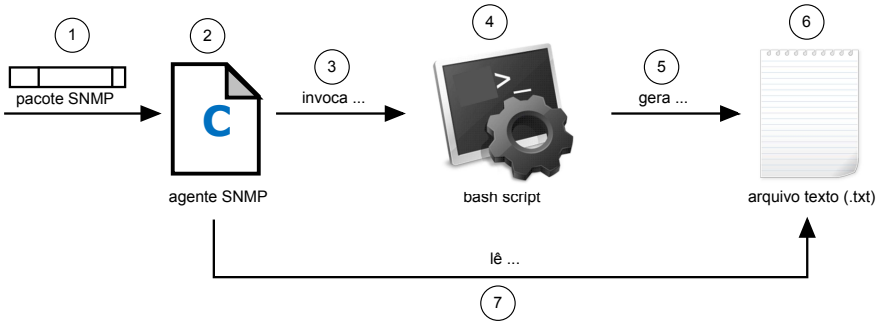


Figura 3.3: Fluxo de operação do agente SNMP implementado.

d) Comunicação interprocessos precisa ser implementada. Na abordagem proposta, o agente opera seguindo um fluxo de ações que passa pela execução do código implementado, chamadas de sistema (novos processos) para acionar *scripts bash* que acionam as APIs dos hypervisors e retorno das informações coletadas ao agente, por meio de arquivos de texto. Outras abordagens podem ser possíveis e estudadas dependendo dos recursos dos sistemas utilizados.

O fluxo de operação do agente encontra-se apresentado na Figura 3.3. *1*) As requisições de entrada, encaminhadas por meio de pacotes do protocolo SNMP são tratadas em etapas (MODE_GET, MODE_SET_RESERVE1, MODE_SET_RESERVE2, MODE_SET_FREE, MODE_SET_ACTION, MODE_SET_UNDO e MODE_SET_COMMIT) *2*) pelo código do agente implementado (virtualRouterMIB.c e vrIfConfigTable.c), de acordo com o tipo de requisição (leitura ou escrita) e com a execução da operação (sucesso ou erro). *3*) O agente então invoca os *scripts bash* necessários para tratar a requisição, *4*) que chamam os métodos da API subjacente do *hypervisor* para executar as ações, *5*) escrevendo as informações de saída resultantes em um arquivo de texto (.txt). *6*) Este arquivo contém um formato (organização) pré-definido, *7*) que é então lido pelo agente e processado (após finalizada a chamada de sistema).

Os seguintes arquivos de código fonte foram desenvolvidos em duas versões para as plataformas VMWare Server e Xen Server, com pequenas diferenças:

- virtualRouterMIB.h: este *header* contém a declaração dos tipos (estruturas) e da interface das funções utilizadas para implementação completa da MIB Virtual Router (Figura 3.2), com exceção dos objetos associados ao grupo vrIfConfig. Além disso contém definições importantes, incluindo o diretório em que o código do agente está localizado (que deve ser editado conforme a utilização);

- virtualRouterMIB.c: este arquivo contém o código fonte para implementação completa da MIB Virtual Router (Figura 3.2), com exceção dos objetos associados ao grupo vrIfConfig. Inclui métodos para a inicialização de escalares e tabelas (recuperando o estado do dispositivo) e para tratar requisições em escrita ou leitura (incluindo a invocação aos *scripts*);

- vrIfConfigTable.h: este *header* inclui a declaração dos tipos (estruturas) e da interface das funções utilizadas para implementação da tabela vrIfConfigTable. Apesar de estruturalmente separado, o código dessa tabela depende de algumas

variáveis definidas em virtualRouterMIB.h (este arquivo encontra-se incluso no *header*);

- vrIfConfigTable.c: neste arquivo são tratadas as requisições associadas aos objetos do grupo vrIfConfig. Na inicialização, são identificados os roteadores virtuais existentes antes de preencher a tabela de interfaces virtuais. A cada operação em escrita, precisam igualmente ser consultadas as informações dos VRs hospedados. O vínculo direto é estabelecido entre as variáveis vrId (da tabela vrConfigTable) e vrIfId (da tabela vrIfconfig table) (Figura 3.2), internamente representados na estrutura interna das linhas (vrIfConfigTable_entry).

O código foi implementado a partir do agente extensível NET-SNMP 5.5 e armazenado em uma pasta dedicada ("... /net-snmp-5.5/agent/mibgroup/ virtual-router-mib") para o agente estendido. Este código foi dividido propositalmente, separando a implementação da tabela das interfaces virtuais (vrIfConfigTable) e gerando um novo arquivo fonte (com seu *header* correspondente) a fim de evitar o acumulo excessivo de código e melhorar a modularidade.

A Tabela 3.1 apresenta as principais características dos *scripts* utilizados nas plataformas Xen e VMware:

Na Tabela 3.1 pode ser observado que alguns *scripts* foram implementados em uma das duas plataformas apenas: o *script* "auth_info.sh" não foi necessário no Xen Server, devido ao fato que o método de acesso à API deste *hypervisor* ("xe ...") exige que seja executado a partir de um usuário com permissões de *"root"*, enquanto que no VMware o usuário é identificado a cada acesso ("vmrun ...-u <usuário> -p <senha> ..."); o *script* "get_vr_by_name.sh" só foi necessário no VMware, devido a forma como são exibidas as informações de cada VM nesta plataforma; os *scripts* "get_if_update.sh" e "go_if.sh" só foram implementados em uma das plataformas por questões de facilidade de uso, não tendo sido utilizados na avaliação experimental. Embora a API das plataformas Xen e VMware sejam diferentes, alguns *scripts* apresentaram grande semelhança na forma de operação, como no caso do "vmstart.sh" e "get_vr.sh". Outros apresentam diferenças significativas, como no caso do "vmcreate.sh", que realiza a criação de um novo VR manualmente por cópia no VMware, enquanto possui um método exclusivo no Xen; o *script* "vmdestroy.sh" possui métodos (formas) distintas de realizar a mesma operação em cada uma das plataformas; por fim, os *scripts* "create_if.sh" e "get_if.sh" são executados editando diretamente arquivos de configuração no VMware, enquanto no Xen são implementados com métodos específicos disponibilizados pela API do *hypervisor*. Como um protótipo para estudo e avaliação, nem todos objetos foram implementados no agente desenvolvido neste trabalho. Em particular, os objetos da tabela de estatísticas ("vrStatTable") poderiam demandar uma ampla quantidade de *scripts* adicionais, porém este esforço de implementação não agregaria resultados aos experimentos realizados.

3.3 Cenários de Gerenciamento

Nesta seção são apresentados cenários de gerenciamento em que a virtualização de redes é empregada (Subseções 3.3.1, 3.3.2 e 3.3.3). Os cenários estudados fornecem uma análise operacional da aplicabilidade e adequação da solução proposta neste

Tabela 3.1: Scripts de acesso às APIs dos hypervisors

Script	Xen Server	VMware Server
auth_info.sh	-	Contém as variáveis nome de usuário, senha e *datastore* padrão.
vmcreate.sh	Cria um novo VR com o nome "router<param1>", onde "param1" é o primeiro parâmetro passado como entrada para o script.	Cria um novo VR com o nome "vr<param1>", onde "param1" é o primeiro parâmetro passado como entrada para o script. Imprime o identificador gerado ("vr uuid") em um arquivo temporário de saída
vmstart.sh	Inicializa o roteador identificado no primeiro parâmetro de entrada para o script.	Inicializa o roteador identificado no primeiro parâmetro de entrada para o script.
vmdestroy.sh	Para abruptamente o VR, desregistra do sistema e apaga a imagem do disco.	Para abruptamente o VR e desinstala do sistema.
get_vr.sh	Lê informação de VRs instalados no sistema e imprime identificadores em arquivo temporário.	Lê informação de VRs e VIs instalados no sistema e imprime identificadores em arquivos temporários.
get_vr_by_name.sh	Lê informação de um único VR identificado pelo primeiro parâmetro de entrada e imprime resultados em arquivo temporário.	-
create_if.sh	Cria uma nova interface virtual editando arquivo de configuração do VR.	Cria uma nova interface virtual utilizando método da API. Imprime identificador gerado em arquivo temporário.
get_if.sh	Lê interfaces virtuais de um VR (<param1> de entrada) a partir de um arquivo de configuração e imprime informações em arquivo temporário.	Lê interfaces virtuais de um VR (<param1> de entrada) a partir de um método da API do *hypervisor* e imprime informações em arquivo temporário.
get_if_update.sh	-	Lê todas interfaces virtuais de cada VR e atualiza/inicializa tabela vrIfConfigTable.
go_if.sh	Conecta ou disconecta uma VI de um VR (*hot pluggin*).	-

trabalho. Nestes estudos, são aplicados os modelos de consolidação introduzidos na Seção 2.3.

3.3.1 Cenário 1: Consolidando Pontos de Presença dos Provedores de Serviços

Atualmente, as redes podem ser vistas contendo três dimensões essenciais: o número de assinantes suportados, a quantidade de largura de banda disponível e o número de tipos de serviços suportados. Estas três dimensões têm um impacto direto na forma como as redes são construídas. No caso dos provedores de serviços, muitas de suas redes não foram originalmente projetadas para lidar com o rápido nível de crescimento, especialmente quando vindo destas três diferentes direções simultaneamente. Para manter o ritmo, os provedores de serviços geralmente realizam *upgrades on-the-fly*, adicionando equipamentos conforme o necessário. Como muitos serviços devem ser gerenciados separadamente, adicionar um novo serviço geralmente significa adicionar uma nova infraestrutura específica para aquele serviço. Como resultado, muitos dos POPs atendendo áreas que tiveram rápido crescimento no número de assinantes e serviços consistem agora de múltiplos dispositivos (dis-

cretos) fornecendo múltiplos serviços (JUNIPER, 2010).

Consolidação da rede (KOLON, 2004) consiste em utilizar roteadores físicos que reúnam a funcionalidade de múltiplos roteadores reais em um mesmo equipamento (seção 2.3). Um exemplo é a "consolidação horizontal na borda da rede" (seção 2.3.1), em que um único roteador físico fornece acesso aos serviços de longa distância (WAN) para múltiplos assinantes através dos POPs dos provedores de Internet (ISPs).

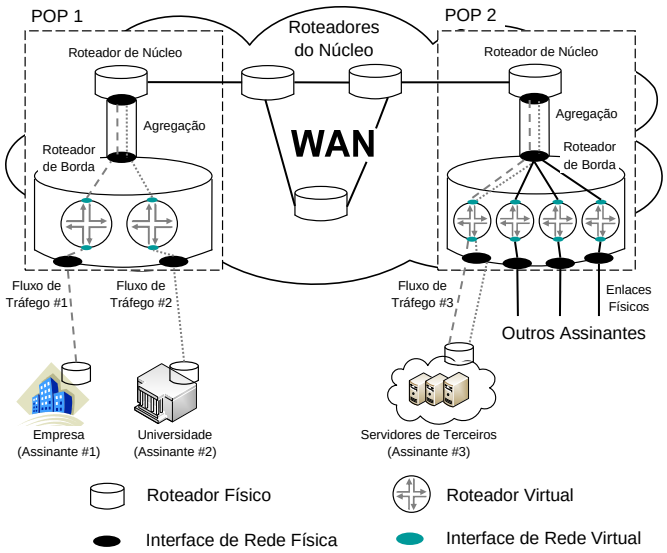


Figura 3.4: Cenário de gerenciamento aplicado à consolidação horizontal na borda da rede.

Na situação suposta na Figura 3.4, onde dois clientes do ISP ("assinante #1" e "assinante #2") precisam acessar um serviço em uma localidade remota (digamos que eles precisem acessar alguma mídia ou estejam utilizando seus servidores em nuvem), são gerados os fluxos de tráfego #1 e #2, respectivamente. Os POPs tendem a adiconar roteadores a fim de suprir novos serviços ou para fornecer conectividade para novos assinantes quando seus dispositivos de borda de rede estão esgotando sua capacidade (JUNIPER, 2010). As novas arquiteturas de roteadores podem lidar com estes cenários utilizando a virtualização. Nesta abordagem, vários roteadores físicos de borda podem ser substituídos por um único dispositivo hospedando vários roteadores virtuais. Com isso, economia real de capital pode ser alcançada com a redução significativa do número de interfaces físicas necessárias para interconectar esses dispositivos ao roteador de convergência do núcleo e entre os próprios roteadores lógicos (conexão inter-roteadores lógicos). O gerenciamento para consolidação horizontal na borda (KOLON, 2004) pode ser implementado por linha de comando (CLI). No entanto, assim que o número de assinantes e o tamanho da rede aumenta, se torna mais difícil capturar e analisar a situação de múltiplos dispositivos a fim de montar corretamente a rede. Dessa forma, o uso de SNMP pode ser uma solução adequada para gerenciar globalmente os roteadores virtuais, disparando configurações e capturando estatísticas dos dispositivos rapidamente.

3.3.2 Cenário 2: Manutenção Planejada

Este cenário apresenta um exemplo de utilização do gerenciamento aplicado no núcleo da rede para realização de uma operação de manutenção em um dos

roteadores físicos da infraestrutura.

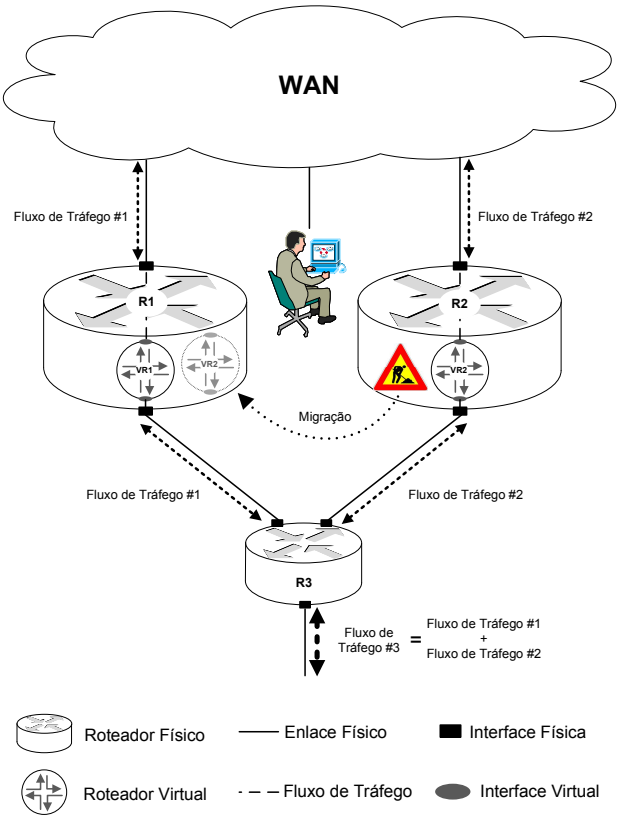


Figura 3.5: Cenário de gerenciamento da migração aplicada no núcleo da rede.

A Figura 3.5 mostra a situação em que o gerente, diante da iminência de manutenção em um de seus roteadores físicos, executa a **"migração"** do correspondente roteador virtual (VR2) para um dispositivo físico próximo (R1). A migração é uma operação que deve ser realizada de forma transparente e com o mínimo de interrupção e perturbação para o funcionamento da rede (WANG et al., 2008). Nesse sentido, a automatização dos procedimentos é essêncial e o gerente deve ficar atento a alguns pontos, em particular:

1. Migração do Enlace (seção 2.1.2): mudança na atribuição de endereçamento da interface física do roteador R3 para que ele transfira o fluxo de tráfego #2 para o enlace que liga R3 a R1;

2. Dimensionamento dos Enlaces: durante a operação normal, o roteador físico R3 poderia realizar o balanceamento automático de carga (ex., executando engenharia de tráfego ou mesmo o protocolo *"WAN Load Balancing"*, fornecido junto ao Vyatta) direcionando o tráfego por ambos os enlaces R3-R1 e R3-R2. No entanto, com a migração, todo o tráfego será direcionado pelo enlace R3-R1. Logo, tanto a capacidade deste enlace (largura de banda) quanto a capacidade do roteador R1 devem ser projetadas para comportar esta potencial demanda.

O uso da migração nos casos de manutenção reduz significativamente o tempo de indisponibilidade da rede, além de facilitar o processo de restabelecimento, diminuindo a necessidade de reconfiguração dos roteadores virtuais e o tempo de reconvergência dos protocolos de roteamento.

3.3.3 Cenário 3: Fornecimento de Múltiplos Serviços

A rede acadêmica SINET3 (URUSHIDANI et al., 2008), no Japão, utiliza os conceitos de "consolidação vertical na agregação/núcleo" da rede (seção 2.3.3) para implementar sua rede WAN de múltiplos serviços. A Figura 3.6 mostra a disposição física e a Figura 3.7 exibe a respectiva representação lógica para esta rede.

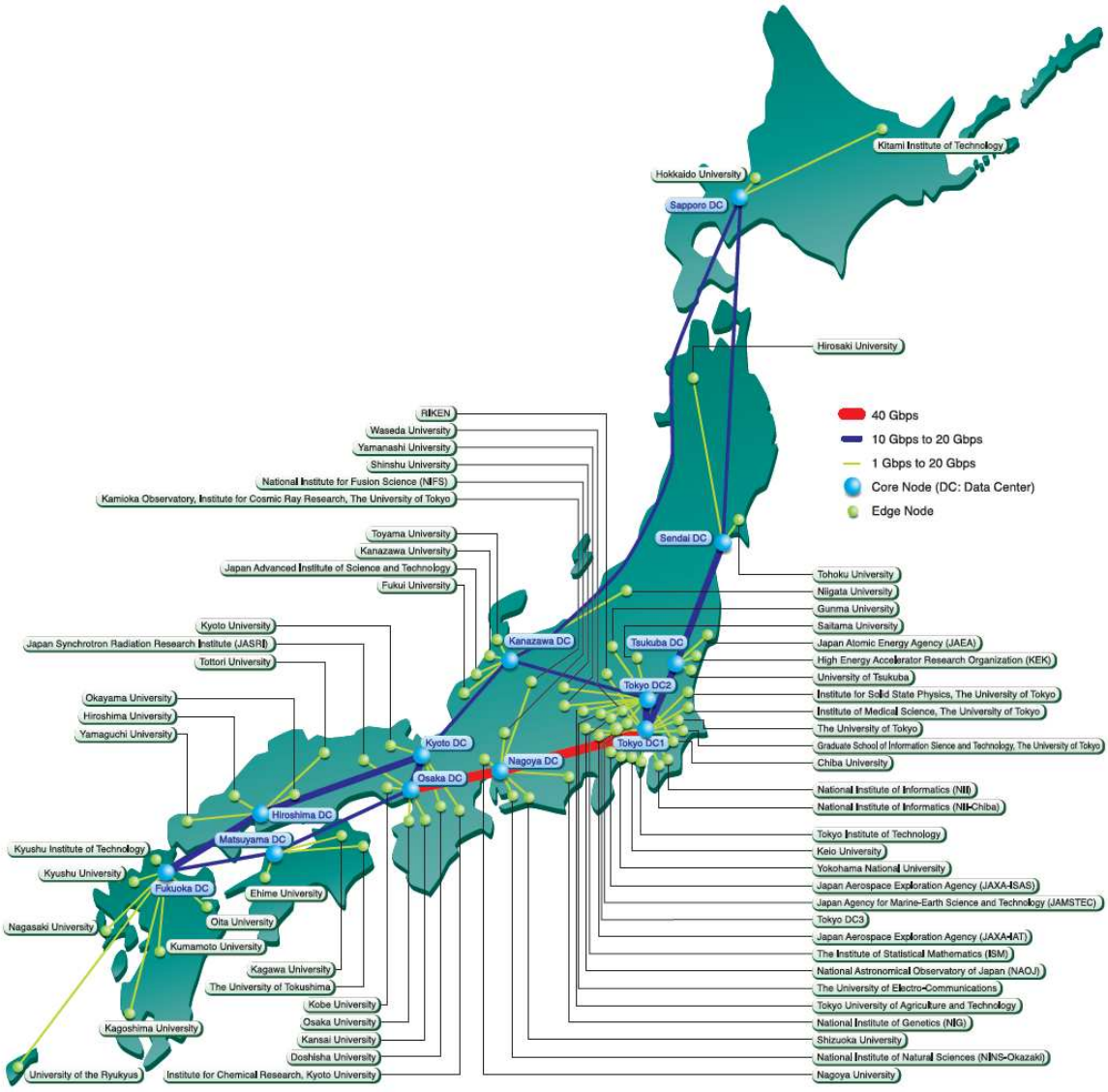


Figura 3.6: Arquitetura física da rede SINET3 (SINET3, 1992).

Para acomodar mais de 700 redes de usuários, a SINET3 possui 63 nodos de borda, que estão localizados nas principais universidades e instituições de ensino (Figura 3.6). A rede também tem 12 nodos no núcleo, que acomodam os nodos de borda em uma topologia em estrela, e possuí rotas reduntantes entre os nodos do núcleo (anéis) para aumentar a disponibilidade dos serviços. A velocidade entre os nodos de borda e núcleo é no máximo de 20 Gbps, e a velocidade entre os nodos do núcleo chega a até 40 Gbps. A rede SINET3 iniciou sua operação em abril de 2007 e utilizou os primeiros enlaces STM256 (40 Gbps) do país entre as cidades de Tokyo, Nagoya e Osaka.

Na SINET3, cada serviço é fornecido na correspondente "rede lógica de serviço" (*logical service network* - Figura 3.8 (a)). Cada uma dessas redes usa seus próprios

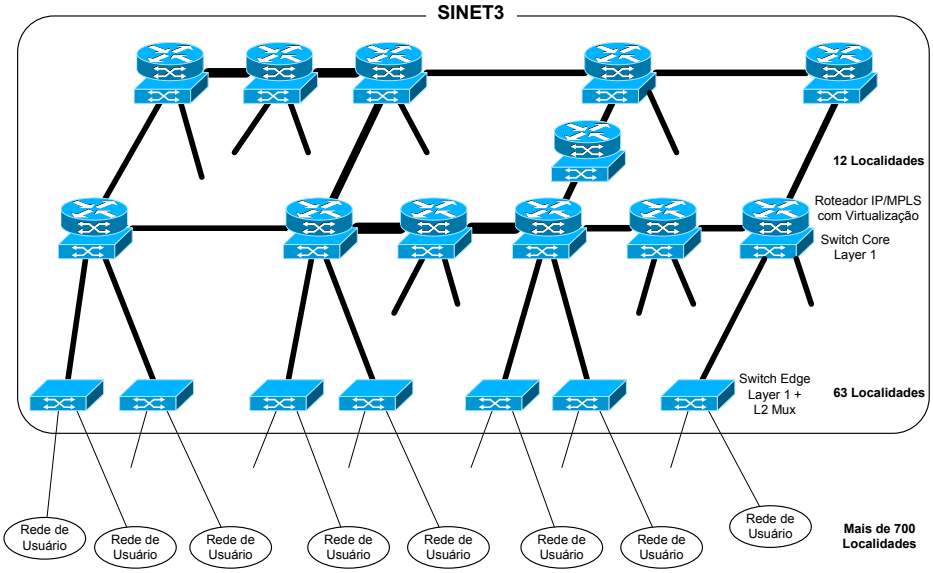
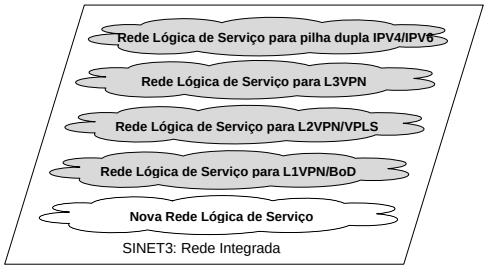


Figura 3.7: Arquitetura lógica da rede SINET3 (URUSHIDANI et al., 2008).

protocolos de roteamento e sinalização para obter informações de roteamento e rótulos para transferir os dados dos usuários. Estão configuradas redes lógicas para os serviços de pilha dupla IPv4/IPv6, L3VPN, L2VPN/VPLS, e L1VPN/BoD. Além disso, o sistema possibilita a criação de novas redes lógicas de serviços conforme as necessidades (Figura 3.8 (b)), utilizando a capacidade de instalar novos roteadores virtuais nos roteadores IP/MPLS do núcleo (Figura 3.7). A criação de redes lógicas de serviços separadas, possibilita o controle dos diferentes protocolos e o gerenciamento das capacidades específicas de cada serviço.



(a) Redes lógicas de serviços (*logical service networks*) (URUSHIDANI et al., 2008).

	Pilha dupla IPv4/IPv6	L3VPN	L2VPN/VPLS	L1VPN/BoD
Sinalização	OSPFv2/v3 e BGP4/4+	RSVP-TE e iBGP, rótulo VPN, OSPFv2	RSVP-TE, iBGP, and OSPFv2	GMPLS (RSVP-TE e OSPF-TE)
Roteamento	Tabela de roteamento incluindo mais de 230.000 rotas	Roteamento individual / tabelas de encaminhamento para as VPNs	Aprendizagem de endereços MAC e tem uma tabela de endereços MAC para o VPLS	-

(b) Descrição das funcionalidades de roteamento e controle de cada rede lógica.

Figura 3.8: Organização da rede SINET3 em redes lógicas (virtuais).

A Figura 3.9 apresenta os principais elementos da rede SINET3. Para conectar as redes de usuários a um nodo de borda (Figura 3.7), os usuários possuem linhas que são fisicamente ou logicamente separadas utilizando VLANS para cada serviço. A Figura 3.9 (à esquerda) mostra o caso em que as linhas dos usuários estão fisicamente separadas por serviço. Um nodo de borda recebe ambos os tráfegos IP (IPv4/IPv6 *dual stack* e L3VPN) e Ethernet (L2VPN e VPLS) sobre interfaces de rede da família Ethernet (10GE/GE/FE), e logicamente os separa anexando os correspondentes tags VLAN internos aos respectivos serviços. O tráfego dos serviços logicamente separados é então acomodado em um caminho óptico SDH compartil-

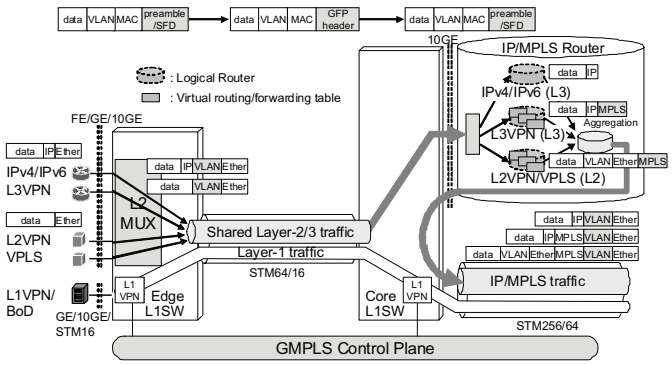


Figura 3.9: Detalhes dos elementos de rede utilizados para acomodar múltiplos serviços (URUSHIDANI et al., 2008).

hado e conduzido até os comutadores de nível 1 do núcleo (*Core L1SW*), conectados aos roteadores de núcelo. Os roteadores IP/MPLS possuem roteadores lógicos para estes serviços, cada um dos quais possuindo capacidades de roteamento, sinalização e encaminhamento independentes (Figura 3.8 (b)). O roteador identifica cada pacote/quadro dos serviços por seu tag VLAN interno e os distribui para um dos roteadores lógicos. Ainda no roteador IP/MPLS, os tags VLAN internos dos pacotes IP são removidos e os dos quadros Ethernet não são removidos. O roteador lógico para a pilha dupla IPv4/IPv6 transfere os pacotes IP como eles estão e os roteadores lógicos para L3VPN/L2VPN/VPLS encapsulam os pacotes IP ou quadros Ethernet com rótulos MPLS. Roteadores IP/MPLS vizinhos estão conectados uns aos outros com interfaces 10 GE via switches layer-1, e os roteadores lógicos vizinhos estão conectados uns aos outros com interfaces lógicas (i.e. VLANs).

Enquanto isso, os serviços da camada 1 (*layer-1*) (L1VPN ou largura de banda sob demanda (*bandwidth-on-demand*)) sobre uma interface 10GE/GE/FE ou STM-16/64 são acomodados em caminhos individuais fim-a-fim da camada 1 (SDH) entre os nodos de borda especificados e estão completamente separados dos outros serviços.

A rede SINET3 é um exemplo particular de aplicação da virtualização de roteadores. A idéia de se utilizar a virtualização para criação de redes lógicas de serviços independentes não é nova (TOUCH et al., 2003). No entanto, a aplicação prática destes conceitos é bastante recente. Uma série de peculiaridades pode ser observada nesta rede, tanto em termos de endereçamento, com a utilização de tags VLAN (diferentemente dos modelos apresentados na Seção 2.3), quanto nos tipos de serviços (caracterizados por tecnologia de encaminhamento das informações - L3VPN, L2VPN/VPLS, L1VPN/BoD - e não por aplicação - voz, video, etc). O modelo de consolidação agregação/núcleo empregado remove a necessidade dos nodos de agregação da rede (Figura 2.8), simplificando a topologia com a utilização apenas das camadas de borda e núcleo. Uma série de casos de uso para as primitivas de gerenciamento apresentadas na seção 3.1 podem ser derivadas deste estudo de caso, como a "criação de uma nova rede lógica de serviços", a migração de enlaces virtuais para o "balanceamento de carga" e a migração de roteadores virtuais para "manutenção de um equipamento").

4 AVALIAÇÃO EXPERIMENTAL

A fim avaliar o impacto do *hardware* sobre o desempenho do agente SNMP implementado, duas plataformas de virtualização distintas foram aplicadas aos cenários de gerenciamento apresentados no Capítulo 3.3: VMware Server (VMWARE, 1998) e XenServer (XEN, 2003). Estes substratos foram utilizados para realizar as configurações necessárias nos rotedores virtuais e fornecer conectividade.

Além disso, é interessante identificar as vantagens e limitações de diferentes soluções no contexto da virtualização de redes. As plataformas de virtualização foram selecionadas de forma que diferentes funcionalidades e paradigmas pudessem ser explorados, como a paravirtualização (no Xen) e a virtualização completa (no VMware). Embora não emulem perfeitamente a virtualização de um roteador real, esses sistemas podem fornecer uma boa compreensão sobre paralelismo, isolamento e configuração dos dispositivos. Como estão disponíveis gratuitamente, experimentos de rede podem ser replicados e redes virtuais podem ser construídas sem a necessidade de adquirir roteadores comerciais que suportem virtualização, os quais geralmente são caros.

Para implementar individualmente cada roteador virtual foram utilizados "roteadores em *software*", que são aplicativos que podem servir para transformar um computador tradicional em um roteador. Esses programas podem ser utilizados em máquinas virtuais nas plataformas de virtualização estudas e são compostos por vários *daemons*, cada um dos quais associado a um protocolo de roteamento como OSPF, RIP, BGP e assim por diante. Desta forma, eles podem implementar completamente o plano de controle e muitas vezes se assemelham ao sistema operacional (IOS) dos roteadores reais. Alguns exemplos incluem "Quagga" (QUAGGA, 2003), "Vyatta" (VYATTA, 2006), "XORP" (HANDLEY, 2004) e "OpenVSwitch" (OPENVSWITCH, 2009).

Neste trabalho, foi escolhido o Vyatta devido a sua facilidade de uso e configuração, documentação disponível e gratuidade. Além disso, o fabricante alega que possui um desempenho comparável a um roteador real (VYATTA-REPLACEMENT, 2010).

As avaliações foram conduzidas em duas etapas:

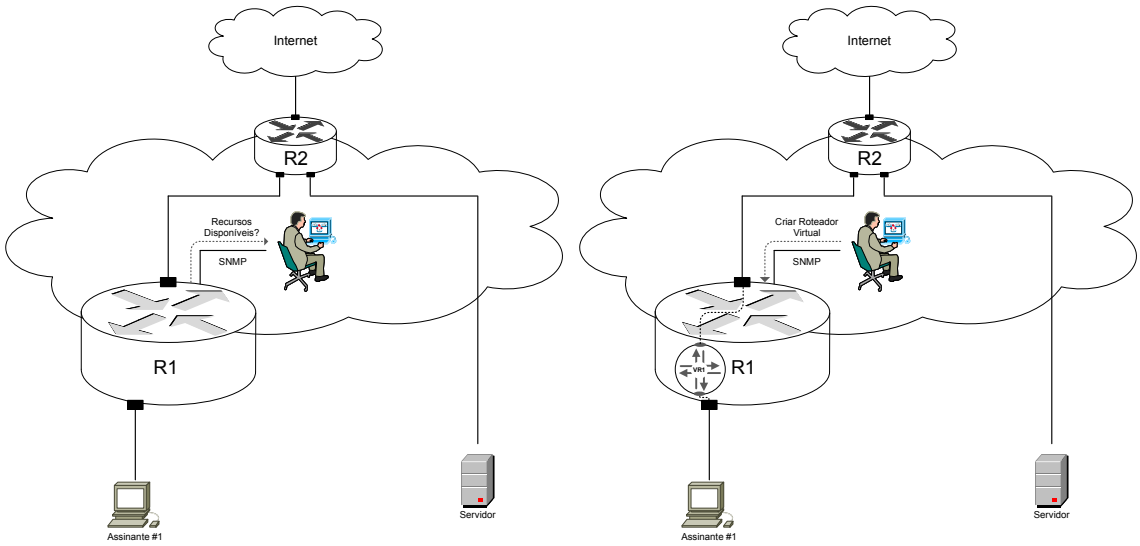
1. Avaliação Operacional: tem o objetivo de fornecer conhecimento e compreensão geral sobre a solução proposta, possibilitando uma análise "qualitativa" sobre a aplicação do agente proposto. Para isso, foi conduzido um caso de uso completo dentro do qual foram selecionadas primitivas de gerenciamento que atendessem à implementação do teste, seguindo uma sequência lógica de execução;

2. Avaliação Computacional: tem o objetivo de fornecer conhecimento sobre o desempenho específico de cada primitiva de gerenciamento, possibilitando uma análise "quantitativa" sobre a aplicação do agente proposto. Para isso, foram selecionadas um conjunto de primitivas de gerenciamento que foram executadas sobre duas plataformas de virtualização distintas.

4.1 Descrição do Experimento 1

O cenário de gerenciamento 1, apresentado na Seção 3.3.1, foi utilizado como referência para criação deste ambiente de teste (*testbed*). Os testes foram conduzidos sobre um dispositivo de borda da rede executando o agente e recebendo tráfego de rede e de gerenciamento. Os procedimentos de investigação foram conduzidos em três passos sobre a plataforma VMware Server:

1. Criando o primeiro VR: neste caso, o substrato não tem carga de uso e possui a capacidade total disponível (Figura 4.1);



(a) Gerente consulta estatísticas do roteador físico.

(b) Gerente dispara o comando SNMP para criação de um novo roteador virtual.

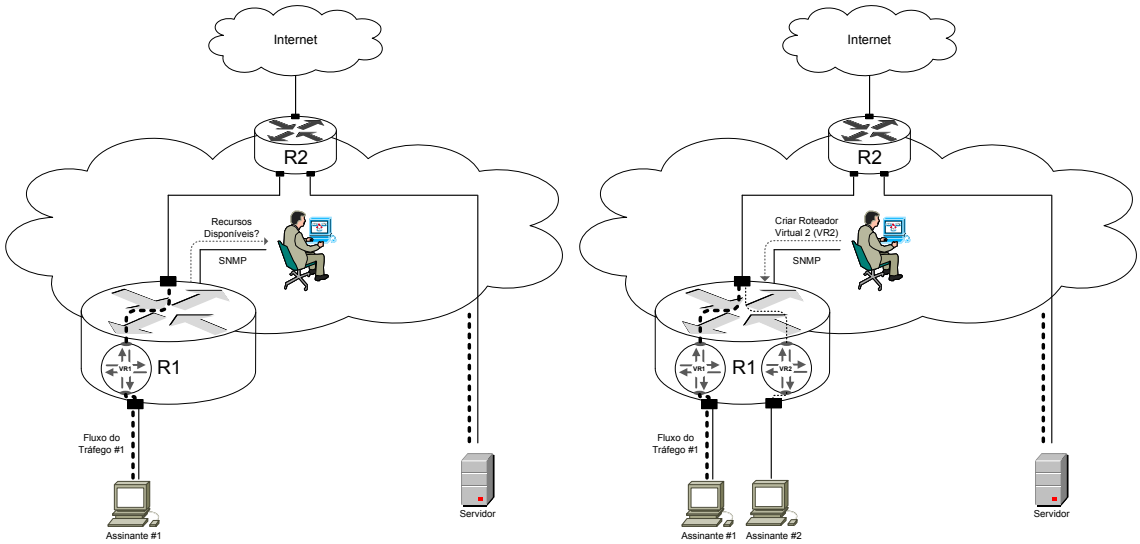
Figura 4.1: Fornecimento de conectividade para acesso do primeiro assinante a rede.

2. Roteando tráfego com o primeiro VR: após a criação, o primeiro VR é colocado em produção enviando/recebendo tráfego de rede do assinante #1 e o roteando para/de um servidor remoto (Figura 4.2 (a));

3. Criando o segundo VR: neste caso, o substrato tem tanto o VR1 instalado quanto sua carga da operação de roteamento (Figura 4.2 (b)).

Em um primeiro momento (Figura 4.1 (a)), nenhum usuário está acessando a rede através de R1. Quando ocorre uma nova demanda, o gerente comanda este roteador para instalar o primeiro roteador virtual (VR1, Figura 4.1 (b)) para que possa ser fornecida conectividade para o assinante #1.

Uma vez conectado, o assinante #1 pode começar a rotear seu tráfego através de VR1 (Figura 4.2 (a)). O administrador de rede do assinante deve migrar os enlaces



(a) Gerente monitorando o estado do sistema.

(b) Gerente aciona a criação de um novo roteador virtual (VR2).

Figura 4.2: Fornecimento de conectividade para acesso do segundo assinante a rede.

ou ativar os dispositivos para direcionar seu tráfego (Fluxo do Tráfego #1) para este novo roteador virtual.

Mais tarde, quando um novo roteador virtual é requisitado outra vez, o gerente deve considerar novos parâmetros que se apresentam, como a carga das interfaces de rede (NICs) e o uso de CPU e memória. Estas estatísiticas de R1 irão determinar a disponibilidade de recursos para instalação do novo roteador virtual (VR2). Finalmente, o gerente pode executar o comando para criar o novo VR de forma segura (Figura 4.2 (b)).

Uma sequência de operações e primitivas de gerenciamento para criação de novos VRs foram identificadas e testadas:

i) Criação do VR;

ii) Criação da Primeira Interface Virtual de Rede;

iii) Criação da Segunda Interface Virtual de Rede;

iv) Comando para Ligar o VR;

v) Tempo de Inicialização do VR;

vi) Configuração do VR;

Neste exemplo, os comandos SNMP de gerenciamento foram executados localmente em R1. Os experimentos foram conduzidos sobre uma máquina com processador Intel® Core^{TM} i7-920 2.66 Ghz e 6 GB de memória RAM (R1), um ponto de acesso (*access point*) D-Link DIR-300 (R2) para interconectar os dispositivos, um *Notebook* convencional e um *Desktop* convencional para transmitir/receber tráfego ("assinante #1" e "servidor"). O tráfego de rede entre os assinantes e o servidor foi gerado com a ferramenta de testes "iperf" 2.0.4 (IPERF, 2008) e os VRs foram

implementados com o roteador em *software* "Vyatta" 6.0. Por fim, para o substrato do roteador R1, foi utilizada a plataforma "Vmware Server" 2.0.2-203138.

Este experimento foi conduzido por meio de intervenções manuais, conforme a metodologia apresentada no Apêndice B.1, sobre a topologia representada nas Figuras 4.1 e 4.2. Primeiramente, foi feito o *logout* da interface gráfica em R1 para minimizar a interferência de processos não relacionados à avaliação. A seguir, o agente SNMP foi inicilizado e os testes executados. Cada teste, individualmente, é disparado em R1 com o script *"scriptXX.sh"*, onde XX é o número do teste que está sendo realizado. Este script bash, por sua vez, inicializa a coleta de informações sobre o uso de CPU e memória e invoca outro script (*"routerY_script.sh"*, sendo Y o número do VR (VR1 ou VR2) sendo manipulado) que dispara a sequência de operações (i a vi) mostrada acima. No último comando, para realizar a configuração de cada VR, é invocado um terceiro script (*"conf_routerZ.tcl"*, onde Z corresponde ao número do roteador virtual VR1 ou VR2), implementado com a ferramenta "Expect" da linguagem TCL, que se conecta via SSH a cada roteador virtual e realiza todas configurações necessárias para conectividade completa entre o Assinante #1 e o Servidor, incluindo as regras de NAT e endereços IP estáticos das interfaces.

Após criado e configurado o primeiro roteador virtual (VR1), o Assinante #1 pode se comunicar com o servidor e vice-versa. Essa comunicação pode ser por meio de mensagens ICMP (via ping) ou com o gerador de tráfego iperf, que utiliza um modelo cliente/servidor: neste exemplo, a aplicação iperf foi inicializada no Servidor para esperar conexões TCP de um cliente, que é executado no *host* do Assinante #1; o cliente pode tanto enviar tráfego no sentido *uplink* apenas quanto enviar tráfego e também se comportar como servidor para que o *host* destino estabeleça uma conexão simultânea (dual) com o *host* origem (Figura 4.2 (a)). Então, o mesmo procedimento de criação de roteador virtual e roteamento de tráfego foi realizado para VR2 e para o Assinante #2, respectivamente (Figura 4.2 (b)). Por fim, o procedimento de teste é finalizado e reinicializado manulamente, apagando as VMs e rodando o script de recompilação do agente (Apêndice A.4). Foram realizadas 10 simulações e considerado um nível de confiança de 95% para cada valor obtido.

4.2 Resultados para o Experimento 1

Nesta seção são apresentados os resultados para os testes realizados no experimento 1, conforme a organização e metodologia descritas na Seção 4.1.

4.2.1 Tempo de Resposta

Os tempos de cada ação necessária para instalação do primeiro e segundo VRs são apresentados na Figura 4.3. O tempo para fornecer conectividade para os assinantes é crítico para os negócios de um ISP, uma vez que este só pode começar a cobrar seus clientes a partir do momento em que o serviço estiver disponível. O tempo medido corresponde ao intervalo entre a requesição (*"SetRequest-PDU"*) e a confirmação (*"GetResponse-PDU"*).

O gráfico da Figura 4.3 mostra que o tempo de execução das ações para instalação dos roteadores VR1 e VR2 foram praticamente os mesmos. Isso indica que não houve interferência da carga do dispositivo (para hospedar e rotear o tráfego de VR1) sobre a solução de gerenciamento utilizada. Essa situação decorre tanto do fato do *hardware* ser suficientemente eficiente quanto da solução de gerenciamento

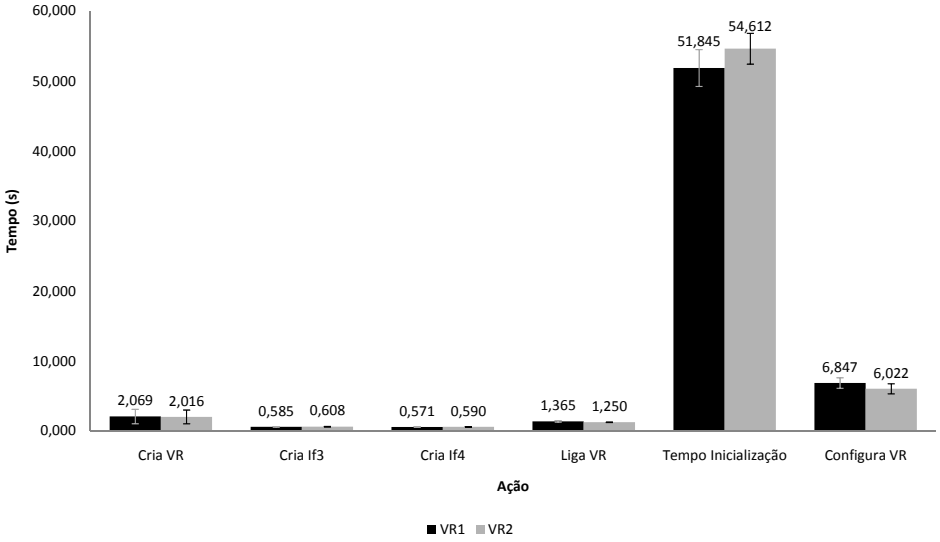


Figura 4.3: Tempo de resposta para cada operação.

ser computacionalmente "leve".

A ação de criação do VR é principalmente baseada no tempo de cópia em disco. Foi observado que re-inicializando o equipamento o tempo aumentava na primeira vez em que o comando era executado e depois disso não modificava-se. Isso ocorre porque na primeira execução a imagem do *template* utilizado na criação de um VR é efetivamente buscada do disco, enquanto nas vezes seguintes em que o experimento era repetido essas informações já estavam na memória e eram buscadas de lá. Por ter sido implementada desta forma, um parâmetro muito importante a ser considerado é o tamanho da imagem do *template* do *software router* utilizado como VR: 692,9 MBytes para implementação do VMware Server.

O tempo de criação das interfaces virtuais de rede (If3 e If4) foi baixo e praticamente constante em todos os casos. Isso ocorre devido a simplicidade desta ação. No VMware ela consiste na edição do arquivo de configuração da VM (escrita em arquivo de texto), seguido por um comando de *refresh* enviado ao *Hypervisor*.

A ativação do VR é feita com o comando de ligar (*active*), disparado via SNMP, seguido pela espera de inicialização. A ação de ligar solicitada ao agente é executada via encaminhamento ao *Hypervisor*, que dispara o comando retornando o *status* da ação, que é então inicializada. A espera de inicialização foi implementada dentro do *script* de teste com o comando nc ("netcat") que verifica se a porta 22 do serviço SSH está ativa. Este foi, significativamente, o pior tempo (crítico) na operação avaliada. Por ser altamente associado à plataforma de virtualização utilizada e não à solução de gerenciamento proposta, está fora do escopo deste trabalho aprofundar a análise ou otimização desta ação. Podem, porém, ser tiradas algumas conclusões importantes em linhas gerais, como, por exemplo, o fato de não valer a pena paralelizar outras operações (como a criação de interfaces virtuais de rede), já que a escala de tempo de execução destas ações pode ser considerada insignificante.

A configuração do roteador virtual, que é realizada por interface de linha de comando (CLI), foi implementada com a ferramenta Expect (LIBES, 1990), uma extensão para linguagem Tcl/Tk. Esta ferramenta permite enviar comandos e receber respostas "esperadas". Com isso, foi automatizado o procedimento de configuração através de um script contendo o acesso SSH seguido por uma sequência de comandos

individuais de configuração (outra opção seria implementar a carga de um arquivo de configuração do dispositivo, de maneira similar).

O tempo de execução automatizada de uma operação como a que foi apresentada é muito menor do que o melhor caso possível de intervenção manual com o mesmo propósito: neste caso a(s) pessoa(s) que iria(m) instalar um novo roteador de borda em um POP, deveria(m) comprar o equipamento, instalar uma interface de rede e então conectar os enlaces deste dispositivo aos roteadores de agregação e dos usuários. Além disso, o operador precisaria inicializar e configurar o novo roteador de borda.

4.2.2 Consumo de CPU

A seguir, foi analisado o uso de CPU ao longo do tempo da sequência de operações (Figura 4.4), com 3 segundos de intervalo de amostragem (comando bash: top -b -d 3). A carga da CPU, assim como a utilização de memória (Seção 4.2.3) são parâmetros decisivos para alocação de novos roteadores virtuais.

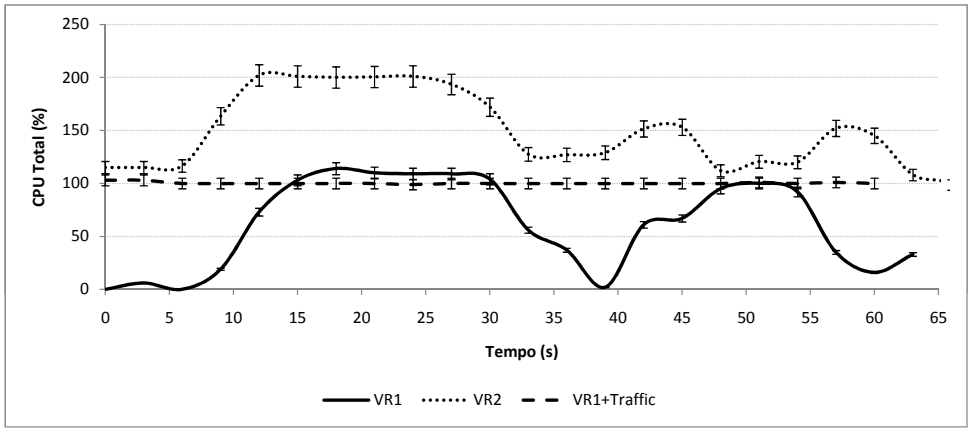


Figura 4.4: Carga total de CPU dos *cores* para as operações sobre os VRs.

A Figura 4.4 mostra a comparação do percentual total de CPU (somatório de cada núcleo, ou *core* de processamento) utilizado durante a criação do primeiro VR (quando o substrato não possui nenhuma utilização - série de dados "VR1"), quando VR1 passa a rotear tráfego (série de dados "VR1+Traffic") e ao longo da criação do segundo VR (quando o roteador já está hospedando e roteando o tráfego do VR1 - série de dados "VR2"). Cargas acima de 100% indicam que múltiplos núcleos de processamento (*multi-core*) estão sendo utilizados. Pode ser observado que o fluxo do tráfego TCP gera uma carga de CPU de fundo praticamente constante de 100% (série "VR1") sobre o VR1, o que significa que a máquina virtual que implementa este roteador virtual está utilizando a capacidade total de um *core*. O elevado uso de CPU para o roteamento não é atípico: de fato, em equipamentos comerciais, esta funcionalidade é implementada com um *hardware* do plano de dados dedicado para esta finalidade (*forwarding fabric*). Nesse sentido, a virtualização agrega algumas funcionalidades adicionais para o gerenciamento e controle do dispositivo, limitando a quantidade de processadores alocados por VR , o que poderia ser utilizado em equipamentos que atuam diretamente na inspeção de pacotes, como *firewalls*, os quais podem ter sua atividade comprometida por ataques do tipo DoS (*"Deny of Service"*).

Também pode ser visto que a carga de CPU oscila durante a criação dos VRs e que é praticamente o dobro do VR1 para o VR2. Estes resultados indicam que, seguindo esta tendência, a plataforma de virtualização escolhida consumiria a capacidade total do sistema quando 8 VRs fossem criados, já que o *hardware* do substrato pode executar até 8 *threads*. Nesta situação, a plataforma não poderia hospedar novos VRs sob risco de interferirem na operação uns dos outros.

4.2.3 Consumo de Memória

As informações de utilização de memória foram coletadas em conjunto com o consumo de CPU (comando "top"), em intervalos de amostragem de 3 segundos.

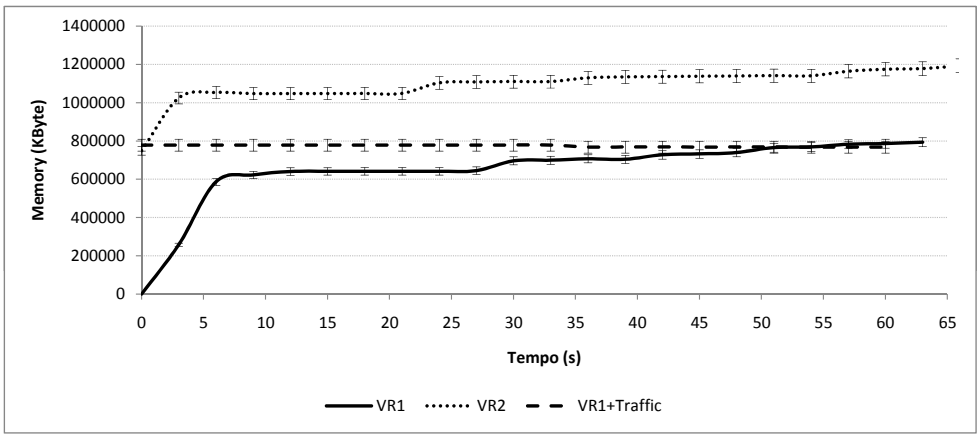


Figura 4.5: Uso de memória total para as operações sobre os VRs.

A Figura 4.5 mostra a utilização total de memória para as operações de criação e configuração dos VRs nos mesmos períodos apresentados nos resultados das Figuras 4.3 e 4.4. Os intervalos analisados correspondem à criação do primeiro VR (curva "VR1"), operação do primeiro VR sob condições de tráfego (curva "VR1+Traffic") e criação do segundo VR (curva "VR2"). Pode ser observado que a alocação principal de memória se dá em fatias que ocorrem nos instantes de criação dos VRs (começo das curvas "VR1" e "VR2"). Este resultado é esperado pela forma como as plataformas de virtualização reservam memória para as máquinas virtuais. Nestes testes, foi configurada a utilização de 512 MB para cada VR.

Na Figura 4.5 pode ser visto também que a utilização de memória das curvas "VR1" e "VR1+traffic" ficou acima dos 512 MB. Na curva VR2, quando a plataforma possui ambos VR1 e VR2 operando, a utilização de memória também ficou acima dos 1024 MB esperados. Isso ocorre devido ao uso da virtualização completa no VMware Server, na qual a plataforma de virtualização compartilha memória com outros processos do sistema operacional subjacente.

A alocação de memória depende no propósito de uso de cada VR e pode ser otimizada individualmente a fim de economizar recursos. É provável, por exemplo, que os VRs 1 e 2 não estejam utilizando, internamente, todos os 512 MB de memória disponíveis para cada um deles. Seria possível replanejar a alocação dessa quantidade de memória conforme as diferentes necessidades de cada VR (ex., um VR que utilize diversos serviços poderia ter mais memória que outro que não precise tantos recursos).

4.2.4 Impacto sobre o Tráfego

O próximo teste analisou as condições de tráfego de um roteador virtual em situação de operação. Intervenções em roteadores que estão em produção são sempre delicadas, já que o impacto sobre os fluxos de dados dos usuários não pode ser previsto. Para este teste, foram aplicados fluxos (*streams*) TCP unicast sobre VR1 (Figura 4.2 (a)) com a ferramenta de teste Iperf. Os dados foram transmitidos através do roteador virtual entre o Assinante #1 e o Servidor (e vice-versa). Foi então medida e feita uma média (também com o Iperf) do tráfego total passando pelo VR em dois momentos distintos: *i*) logo após a criação do primeiro VR (Figura 4.2 (a)), em um intervalo de tempo em que o substrato não possui interferência externa por outros eventos; e *ii*) durante a criação do segundo VR, em um intervalo de tempo equivalente a (i), porém com a medição realizada durante o período de criação do segundo VR (VR2).

Tabela 4.1: Tabela de tráfego TCP

	VR1: Operação regular		VR1: Durante Instalação de VR2	
Sentido	**Unidirecional (Mbits/seg)**	**Dual (Mbits/seg)**	**Unidirecional (Mbits/seg)**	**Dual (Mbits/seg)**
Uplink	93,8	93,2	94,0	92,5
Downlink	74,8	19,8	74,5	20,1

A Tabela 4.1 apresenta a média da vazão ao longo do intervalo de tempo de execução da sequência de operações para instalação de um novo roteador virtual (VR2), a fim de avaliar o impacto sobre o tráfego roteado pelo roteador virtual em atividade (VR1). Nesta tabela, a segunda e terceira colunas exibem os valores do tráfego medido durante a operação regular de VR1 e a quarta e quinta colunas durante a criação de VR2.

Inicialmente, foi medida a máxima vazão disponível em VR1 para o tráfego unidirecional *uplink* (do assinante #1 para o servidor) e *downlink* (do servidor para o assinante #1). Existe uma diferença de vazão entre os dois sentidos devido ao fato de no caso uplink estar sendo utilizado NAT convencional a partir da origem (*source NAT*) enquanto no sentido downlink estar sendo utilizado NAT por encaminhamento de porta (*port forwarding*) a partir do destino (*destination NAT*). A seguir, foi avaliada a transmissão simultânea (dual, ou *full duplex*). Estes resultados estão apresentados na segunda e terceira colunas. Por fim, foram avaliados os mesmos fluxos de tráfego durante a criação do VR2. Estes testes mostram que a operação de criação e instalação (*deploy*) do segundo roteador virtual foi praticamente imperceptível para o primeiro roteador virtual, do ponto de vista da média de vazão ao longo do intervalo de criação. A capacidade do *hardware* utilizado é muito importante neste sentido, uma vez que possibilita o gerenciamento em tempo de execução (sem a necessidade de desabilitar o serviço de outros assinantes). Estes resultados, no entanto, não significam que não exista qualquer interferência no fluxo do tráfego #1.

A Figura 4.6 mostra o comportamento detalhado dos tráfegos roteados por VR1 e VR2 durante a sequência de operações do experimento 1, dividida em intervalos. O traço da Figura 4.6 foi escolhido aleatoriamente entre um conjunto de resultados. O experimento não é perfeitamente replicável, devido aos diferentes tempos de ex-

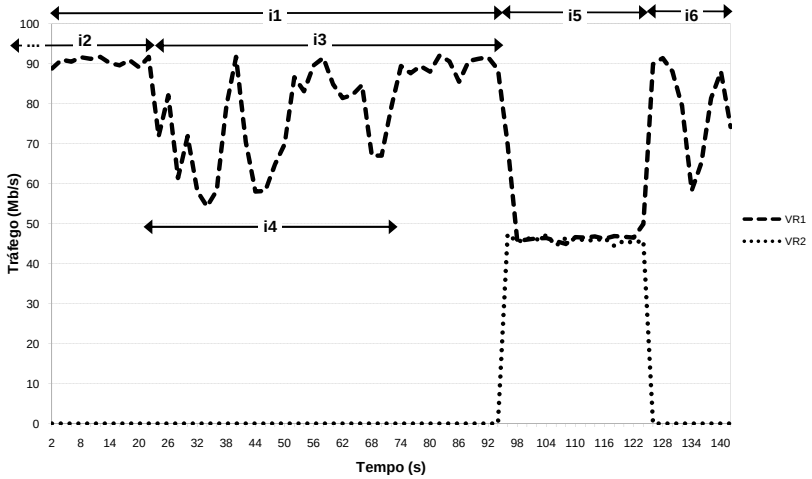


Figura 4.6: Traço de fluxo dos tráfegos durante a criação de um novo VR (VR2).

ecução de cada ação, mas possui similaridades recorrentes que evidenciam a avaliação apresentada. Por restrições do *hardware*, os assinantes #1 e #2 da Figura 4.2 (b) foram implementados em um mesmo host com duas máquinas virtuais, compartilhando um único enlace até o roteador R1.

O intervalo "i1" inicia logo após VR1 ser inicializado, começando a rotear o tráfego do assinante #1, e se estende até o momento em que, após criado e incializado, VR2 começa a rotear tráfego do assinante #2. O intervalo "i5" inicia quando VR2 começa a rotear seu tráfego, compartilhando o enlace com VR1 e termina quando VR2 é desligado. O compartilhamento pode ser observado pela redução da vazão de cada VR pela metade (aproximadamente 47 Mb/s). O intervalo "i6" começa quando VR2 é finalizado e se estende até o final da remoção completa deste dispositivo. Como pode ser visto, a remoção voluntária de VR2 faz com que VR1 recupere sua vazão original (aproximadamente 94 Mb/s), porém a operação completa de remoção de VR2 (parar, desregistrar VM, apagar VM do disco, etc.) impacta no fluxo de VR1. O intervalo "i2" corresponde ao tempo em que, após inicializar VR1, este equipamento fica operando sozinho roteando tráfego de seu(s) assinante(s). Neste intervalo não existem interferências externas e o roteador virtual pode atingir sua capacidade máxima de transmissão unidirecional *uplink* (93,8 Mbits/s, na média, conforme a tabela 4.1). O intervalo "i3" corresponde ao tempo ao longo do qual transcorrem as ações apresentadas no gráfico da Figura 4.3 para o VR2. Neste período, a transmissão unidirecional *uplink* do roteador virtual foi, na média, de 94,0 Mbits/s, conforme a tabela 4.1. Os períodos de amostragem "i2" e "i3" para levantamento da vazão média de VR1 e VR2 estabelecidos foram bastante próximos e suficientemente longos para garantir a confiabilidade dos dados e da comparação. Por fim, o intervalo "i4" indica o período no qual o tráfego de VR1 foi mais impactado pelas ações de gerenciamento executadas sobre VR2, que correspondem, pelos tempos da Figura 4.3 a criação do VR, criação das interfaces virtuais, ligar e incialização do VR (a configuração, em si, praticamente não impactou no desempenho).

Dessa forma, existe uma interferência nos tráfegos dos roteadores virtuais que deve ser mitigada no projeto do *hardware* do dispositivo para viabilizar o gerenciamento seguro em escalas maiores de equipamentos e dispositivos virtualizados. O "Juniper Networks JCS1200 Control System" é uma abordagem comercial de equipa-

mento que adota a estratégia de desacoplar o plano de controle dos equipamentos para um dispositivo dedicado, possibilitando a criação de roteadores virtualizados em *hardware* que garantem maior desempenho e separação dos recursos.

4.3 Descrição do Experimento 2

Neste experimento, foram abordados aspectos computacionais de duas plataformas de virtualização distintas, buscando estressar as capacidades dos dispositivos para avaliar o desempenho das primitivas de gerenciamento estudadas neste trabalho.

Os testes foram conduzidos sobre o mesmo ambiente de teste apresentado na seção 4.1, porém o equipamento de borda apresentado foi operado por duas plataformas de virtualização concorrentes (Citrix Xen Server e VMware Server) instaladas em discos rígidos separados e testadas em momentos alternados. Não é objetivo, no entanto, aprofundar a análise a detalhes específicos da implementação de cada plataforma, mas sim fornecer uma avaliação comparativa da solução de gerenciamento aplicada a diferentes substratos.

Nesse estudo, as avaliaçãoes foram feitas considerando a criação de sucessivos VRs. Cada novo VR n foi criado sobre o mesmo roteador físico hospedando os $n-1$ VRs anteriores. O número n de VRs varia de 1 a 10. Os resultados foram testados 30 vezes e foi considerado um nível de confiança de 95%.

Este experimento foi conduzido de forma totalmente automática, por meio do *script "case10.sh"*, conforme a metodologia apresentada no Apêndice B.2. Os testes foram realizados sobre a topologia representada nas Figuras 4.1 e 4.2, com pequenas adaptações nas versões para as plataformas VMware Server e Xen Server (configurando o caminho das pastas dos *scripts* de teste (variável "LOCAL_FOLDER") e do agente NET-SNMP (variável "NET_SNMP_FOLDER") no arquivo de configuração dos testes, *"auth_info.sh"*). Basicamente, o *script "case10.sh"* dispara 30 interações (testes) com a seguinte sequência de etapas:

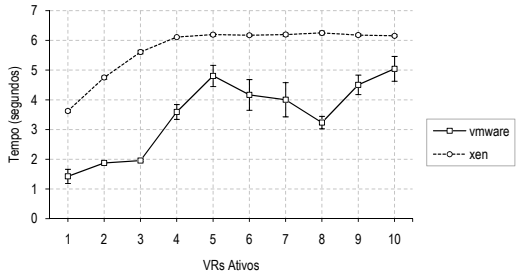
1. Inicializar agente SNMP;
2. Repetir 10 vezes (casos) os seguintes passos:
   (a) Medir utilização de CPU e Memória;
   (b) Invocar o *script "router_script.sh"* para executar e medir o tempo de resposta das primitivas de criação do VR, criação das VIFs e ativação do VR;
3. Repetir 10 vezes (casos) os seguintes passos:
   (a) Invocar o *script "removal_script.sh"* para executar e medir o tempo de resposta da primitiva de remoção do VR;
4. Finalizar medição do uso de CPU e Memória.

Dessa forma, os 10 VRs são primeiramente criados em sequência e então removidos, também em sequência. Nenhum tráfego de rede foi aplicado a estes testes. Embora alguns testes sejam similares, os resultados deste experimento não devem ser comparados aos obtidos para o experimento 1, já que neste caso as operações

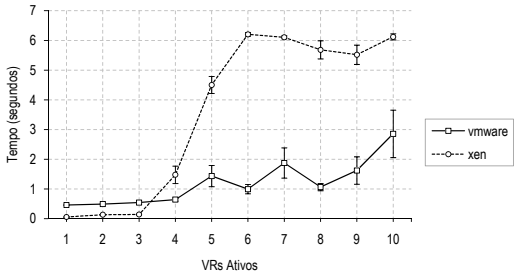
são disparadas em rajadas (ex., a criação de um VR possivelmente ocorre durante a inicialização de algum outro VR) enquanto no primeiro experimento os testes foram realizados pausadamente, aguardo a finalização completa de cada operação.

4.3.1 Tempo de Resposta

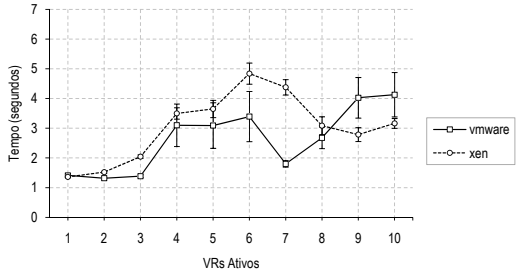
Foi computado o atraso médio, observado do lado do gerente, para as operações de criação de um VR, criação de uma interface virtual de rede, inicialização de um VR, e remoção de um VR (Figura 4.7). O tempo medido corresponde ao intervalo entre a requesição (*"SetRequest-PDU"*) e a confirmação (*"GetResponse-PDU"*).



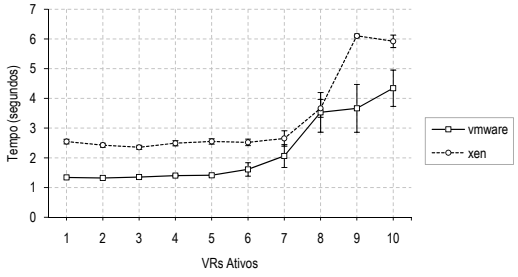
(a) Tempo de resposta para criação de VRs



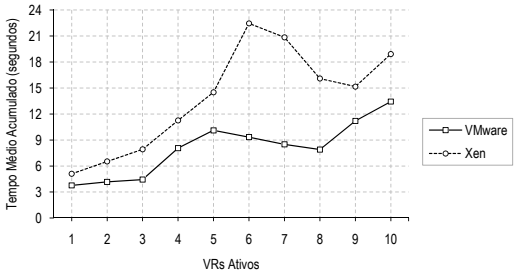
(b) Tempo de resposta para criação de uma interface virtual de rede



(c) Tempo de resposta para ativação de um VR



(d) Tempo de resposta para remoção de um VR



(e) Tempo acumulado para as operações sobre VRs

Figura 4.7: Tempo de resposta para as operações sobre VRs

Os gráficos da Figura 4.7 mostram o tempo (eixo Y), medido em segundos, para um número de roteadores virtuais instalados variando de 1 a 10 (eixo X). Para criação de VRs (Figura 4.7(a)) o tempo de resposta tende a se estabilizar depois da criação do quarto VR no Xen. No VMware, existem variações nos tempos obtidos para um número diferente de VRs ativos, porque, neste caso, o consumo (ex., CPU e memória) varia mais frequentemente ao longo do tempo. Embora o tempo tenha sido relativamente próximo para ambas as plataformas, a forma de implementação desta operação para cada uma delas foi bastante diferente. Enquanto

no VMware, a criação de um novo VR foi feita copiando e colando diretamente os arquivos, e então registrando a nova VM na plataforma (‘‘vmrun -T server -h https://localhost:8333/sdk -u $HOST_USER -p $HOST_PASS register "[$DATASTORE_NAME] router$1/router.vmx"’’), no Xen foi utilizado um comando interno específico para clonagem de imagens (‘‘xe vm-clone vm=vyatta-virtualrouter new-name-label=vr$1’’).

Para criação de uma interface virtual de rede (Figure 4.7(b)) o Xen foi melhor que o VMware para 3 ou menos VRs ativos. Para 4 ou mais VRs, o VMware apresentou um tempo de resposta muito menor. O tempo de criação de interfaces virtuais de rede no Xen tende a estabilizar entre 5 e 6 segundos para 5 ou mais VRs ativos. Esta operação também foi implementada de forma distinta entre as plataformas. No VMware foi feita editando diretamente os arquivos de configuração da VM de cada VR e a seguir informando as alterações à plataforma (‘‘vmware-vim-cmd -U $HOST_USER -P $HOST_PASS vmsvc/reload $2’’), enquanto no Xen foi utilizado o comando interno “vif-create” (‘‘xe vif-create vm-uuid=$1 device=$2 network-uuid=$3’’).

A ativação de um VR é a instrução que aciona a inicialização de um VR, não incluindo o tempo desta etapa (*“boot time”*). Para esta operação (Figura 4.7(c)), o VMware foi mais rápido que o Xen até 7 VRs ativos. Depois disso, o Xen desempenhou a ação mais rapidamente que o VMware. Em ambos os casos foi utilizado um comando interno para disparar esta operação (‘‘vmrun -T server -h https://localhost:8333/sdk -u $HOST_USER -p $HOST_PASS start "[$DATASTORE_NAME] router$1/router.vmx"nogui’’, no VMware, e ‘‘xe vm-start vm=vr$1’’, no Xen).

A remoção dos VRs ocorre após a criação e ativação dos 10 VRs, em sequência inversa (do VR10 para o VR1). Na operação de remoção (Figura 4.7(e)), o VMware apresentou um desempenho superior ao Xen até 7 VRs, equivalente em 8 VRs e inferior a partir de 9 VRs. No VMware, foram executados os comandos internos de parar (‘‘vmrun -T server -h https://localhost:8333/sdk -u $HOST_USER -p $HOST_PASS stop "[$DATASTORE_NAME] router$1/router.vmx"hard’’) e desregistrar (‘‘vmrun -T server -h https://localhost:8333/sdk -u $HOST_USER -p $HOST_PASS unregister "[$DATASTORE_NAME] router$1/router.vmx"’’) e o comando externo (‘‘rm -rf $DATASTORE/router$1’’) para remover os arquivos de cada VR (configurações e imagem), enquanto no Xen foram apenas utilizados os comandos internos de parar (‘‘xe vm-shutdown vm=vr$1 force=true’’) e desinstalar (‘‘xe vm-uninstall vm=vr$1 force=true’’) cada VR.

Por fim, o gráfico da Figura 4.7(e) apresenta o tempo total acumulado para as operações de criação de VRs, criação de uma interface virtual de rede, ativação de um VR e remoção de um VR, correspondente ao somatório dos tempos medidos individualmente nas Figuras 4.7(a), 4.7(b), 4.7(c) e 4.7(d). Aqui, pode ser observado que o VMware foi sempre mais rápido que o Xen para o conjunto de operações como um todo. Curiosamente, para o Xen, o tempo mais alto medido foi para 6 VRs ativos e não para 10, que é quando o dispositivo possui maior carga.

De uma forma geral, é possível observar que as operações sobre os VRs foram executadas com melhor desempenho (menores tempos) no VMware Server que no Xen Server. Apesar da implementação interna das operações pelos *hypervisors* das plataformas VMware e Xen Server não serem conhecidas, pode ser concluído que, embora a implementação interna garanta melhor encapsulamento e acesso para o

usuário, o desempenho das operações não é necessariamente otimizado, já que em muitos casos conseguimos tempos mais baixos implementando as operações externamente (via sistema operacional) no VMware.

4.3.2 Consumo de CPU

A utilização de CPU para as operações analisadas no experimento 2, medida com 2 segundos de intervalo de amostragem, é apresentada na Figura 4.8.

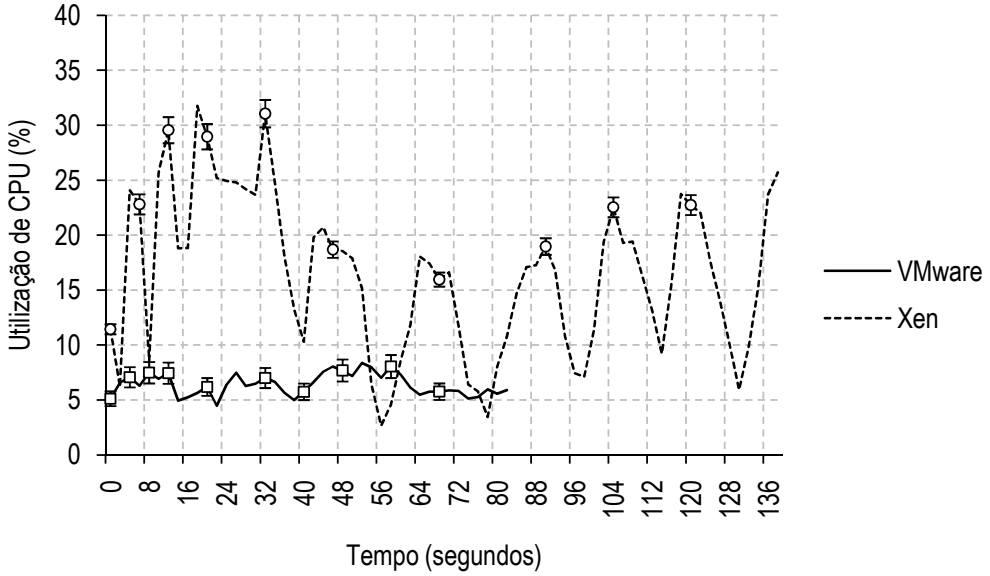


Figura 4.8: Carga de CPU.

Os pontos marcados nas curvas representam os instantes em que um novo VR é criado e colocado em operação (sequência de operações de criação do VR, criação de uma interface virtual de rede e ativação do VR). Os 10 pontos apresentados, correspondem, em sequência aos roteadores virtuais 1 a 10. A diferença no número de amostras plotadas para o VMware com relação ao Xen advém do fato de que, como mostrado na Seção 4.3.1, no Xen a duração das operações é geralmente maior do que no VMware. Assim, o tempo dos testes no Xen também é mais alto e, logo, a quantidade de amostras é maior.

O percentual de CPU apresentado na Figura 4.8 corresponde a fração de utilização sobre o total de processamento disponível no *hardware* adotado. Nestes testes, foi utilizado um processador Intel® Core^{TM} i7-920 2.66 Ghz, que possui 4 núcleos com 2 *threads* de processamento cada. Os resultados indicam que o VMware teve menor consumo de CPU que o Xen. Isso não é necessariamente somente devido às operações dos VRs, já que existem outros processos sendo executados pelo SO da plataforma que podem influenciar estes valores.

Desses resultados é possível concluir que o fato do consumo de CPU ser maior no Xen influencia nos tempos de resposta (Figura 4.7) para execução das operações sobre os VRs nesta plataforma.

4.3.3 Consumo de Memória

Também foi observada a utilização de memória para as operações sobre os VRs, amostrada em conjunto com a utilização de CPU (Seção 4.3.2) em intervalos de 2 segundos (Figura 4.9).

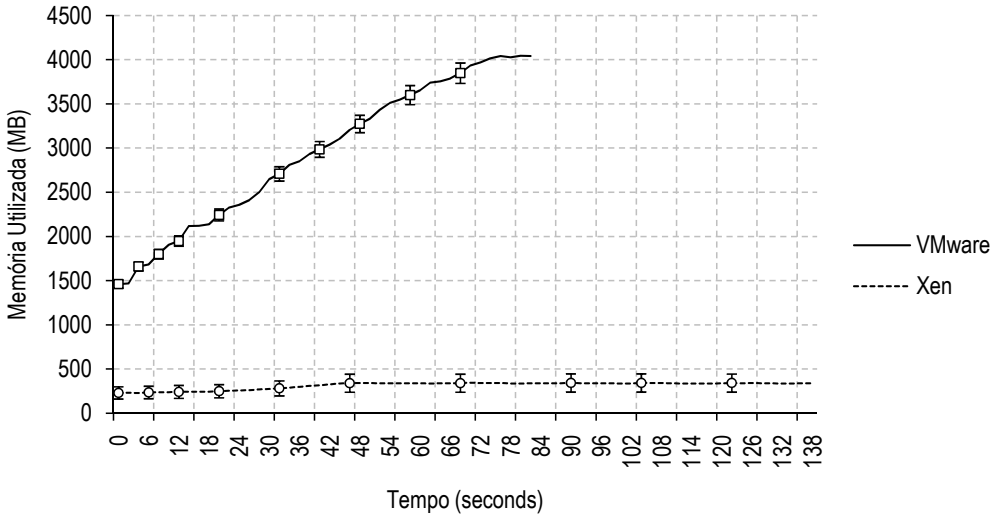


Figura 4.9: Memória utilizada.

Novamente, os pontos marcados nas curvas representam os instantes em que um novo VR é criado e colocado em operação (sequência de operações de criação do VR, criação de uma interface virtual de rede e ativação do VR).

Foram configurados 256 MB de memória para cada VR. O gráfico da Figura 4.9 mostra que a plataforma VMware possuia uma utilização inicial de aproximadamente 1,5 GB de memória que, com a criação dos 10 VRs, aumentou para aproximadamente 4 GB, conforme esperado. No caso do Xen, no entanto, a criação dos VRs não refletiu a alocação estática da memória para cada VM. A plataforma Xen possuia uma utilização incial de 256 MB de memória que se manteve ao longo do tempo, mesmo com instalação dos 10 VRs. Dessa forma, podemos concluir que o Xen utiliza um mecanismo de alocação dinâmica de memória, diferentemente do que ocorre no VMware. Na verdade a alocação dinâmica de memória também pode ser configurada pelo administrador no VMware, mas não é recomendada por questões de desempenho.

Enquanto a utilização de memória no Xen foi significativamente menor que no VMware, a utilização de CPU foi maior (Seção 4.3.2). Dos resultados obtidos é possível concluir que o tempo de resposta (Seção 4.3.1) para as operações sobre VRs é mais sensível aos níveis de carga de CPU do dispositivo em um dado momento do que da utilização de memória. Mais do que isso, poderiamos inferir que o mecanismo de gerenciamento de memória empregado pelo Xen Server, garantiu economia de memória a custa de maior carga sobre a CPU.

5 CONCLUSÃO

Nesta dissertação foram apresentados os principais conceitos e aspectos relacionados às interfaces de gerenciamento para dispositivos de rede com suporte à virtualização. A falta de um padrão comum de interface para a administração de redes virtuais dificulta a operação e o desenvolvimento de ferramentas de gerenciamento para estas redes. Para tratar estes problemas, foi proposta uma solução de gerenciamento baseada no protocolo SNMP e numa base de informações de gerenciamento (MIB) específica, contemplando as principais operações consideradas essenciais. Foram realizados testes usando estudos de caso e coletadas informações para análise da solução. A fim de testar e avaliar a proposta apresentada, foi implementado o protótipo de um agente SNMP para duas plataformas de virtualização distintas, fornecendo resultados para a avaliação qualitativa e quantitativa do trabalho desenvolvido.

5.1 Contribuições desta Dissertação

As principais contribuições deste trabalho são quatro. Primeiro, foi realizada uma revisão sobre o estado-da-arte em virtualização de redes apresentando os principais conceitos relacionados ao gerenciamento deste tipo de ambiente. Segundo, foi identificado e selecionado um conjunto de operações fundamentais definidas como primitivas de gerenciamento, a partir das quais é possível realizar todas as demais operações. Como exemplo, foi elaborado um experimento para realizar a operação de "instalar de um novo VR para um cliente em um ISP"por meio da composição de um conjunto de primitivas. Terceiro, foi desenvolvida uma interface de gerenciamento, acessando de forma padronizada informações de diferentes plataformas de virtualização, que implementam suas funcionalidades com APIs proprietárias específicas de cada fabricante. Quarto, a realização dos estudos de caso (cenários de gerenciamento) e experimentos embasou uma série de conclusões importantes. No plano qualitativo, foi observado que podem aparecer ações ou efeitos derivados a partir da execução das primitivas de gerenciamento, como reflexo. Quantitativamente, foram avaliados os parâmetros tempo de execução, consumo de CPU, consumo de memória e impacto no tráfego de outros VRs instalados. Foram coletados e comparados valores de referência para duas plataformas.

Os resultados obtidos são bastante positivos. A utilização da interface de gerenciamento e da virtualização evidencia a redução significativa no tempo de execução e custos das operações que são tradicionalmente realizadas por intervenção manual de operadores da rede. Além disso, é possível criar scripts e ferramentas mais complexos para automatizar certas rotinas. O *hardware* multicore utilizado nos testes

apresentou desempenho suficiente e compatível com os experimentos realizados. Os valores para os parâmetros tempo de execução, consumo de memória e consumo de CPU se mostraram consideravelmente baixos, mesmo para um grande número de VRs. A interface de linha de comando (CLI) da plataforma Citrix Xen Server, em particular, apresenta uma API bastante completa e adequada para o acesso aos recursos do *hypervisor*. Com a tecnologia SNMP foi possível implementar o agente estudado de forma flexível e modular, por objetos, além de apresentar baixo consumo computacional e de tráfego de rede.

5.2 Considerações sobre a Solução Proposta e Resultados Obtidos

Nesta seção, é fornecida uma análise crítica da solução proposta, destacando seu relacionamento com os conceitos associados, escopo e aplicabilidade. Também são discutidas as principais conclusões obtidas com os resultados dos experimentos realizados.

Neste trabalho, foi apresentada uma interface de gerenciamento para roteadores virtuais. O agente SNMP desenvolvido foi implementado individualmente para cada plataforma (VMware e Xen Server) e a comunicação com os demais roteadores virtuais (Vyatta) foi feita utilizando múltiplos agentes. Nessa abordagem, a administração dos elementos físico e lógicos é feita de forma independente. Nos experimentos realizados, o agente SNMP foi instalado para gerenciar o *hardware* e agentes SSH foram instalados para acessar as informações e configurações de cada roteador virtual. Embora possa parecer haver alguma redundância ou sobrecarga com essa distribuição, a organização com múltiplos agentes favorece o gerenciamento sobreposto. Como o proprietário de um roteador virtual costuma ser uma pessoa ou entidade diferente do proprietário do roteador físico que abriga o roteador virtual, é importante que as funções de gerenciamento do roteador virtual possuam algum mecanismo de proteção para que não afetem o roteador físico associado, isto é, deve existir isolamento entre as funções de gerenciamento do roteador virtual e físico, bem como entre roteadores virtuais de redes virtuais distintas.

A definição de um conjunto de operações que elucidam as peculiaridades da virtualização aplicada às redes faz do estudo de seu gerenciamento uma disciplina particular. Nesse sentido, o gerenciamento de redes virtuais transpassa o gerenciamento das redes convencionais, utilizando uma série de recursos voltados exclusivamente para aspectos operacionais intrínsecos à virtualização, e outros recursos comuns a qualquer rede. Em relação ao escopo da solução, pode ser dito que a interface de gerenciamento proposta aborda um subconjunto básico de funcionalidades para virtualização de roteadores, mas não cobre todos aspectos relacionados às diversas áreas de pesquisa (Seção 5.3), como o controle segregado de acesso entre InPs e SPs, QoS fim-a-fim e o tratamento de falhas, por exemplo.

Do ponto de vista da aplicabilidade, é importante destacar que a implantação completa de um VR foi testada conforme o primeiro cenário de gerenciamento, sendo executada em poucos segundos. Em relação às intervenções feitas tradicionalmente de forma manual, a interface de gerenciamento apresentada reduz não apenas o tempo, mas também os custos, espaço e mão-de-obra envolvida. Além disso, outros estudos de caso com situações diversas de aplicação para solução proposta foram apresentados no Capítulo 3.3. O emprego da interface estudada em ambientes maiores

e mais complexos não exclui a possibilidade de se apresentarem situações inesperadas, não cobertas pela interface, mas a inclusão incremental de novos objetos pode ser implementada facilmente na MIB quanto no agente SNMP, expandindo suas capacidades.

Com relação aos resultados obtidos, o tempo para execução das operações estudadas foi baixo, não ultrapassando 8 segundos, mesmo quando o *hardware* foi submetido a elevadas cargas. Os experimentos mostraram, porém, o aparecimento de efeitos derivados no fluxo de execução do caso de uso avaliado no experimento 1, com tempo bastante acima das demais operações, evidenciando que otimizações apenas sobre o agente desenvolvido melhorariam o desempenho da solução final.

Além disso, as primitivas de gerenciamento estudadas mostraram exercer influência sobre o tráfego roteado pelos VRs (Figura 4.6). Porém, o impacto não chegou a causar alteração significativa sobre a vazão total média (Tabela 4.1). Como os tráfegos mais sensíveis à latência e *jitter* (ex., voz e vídeo) são roteados no plano de dados, isso indica que a interface de gerenciamento desenvolvida neste trabalho não compromete a operação do equipamento.

Dos resultados obtidos foi possível concluir, ainda, que o tempo de resposta para execução das primitivas de gerenciamento estudadas é mais sensível aos níveis de carga de CPU do dispositivo, em um dado momento, do que da utilização de memória. Este resultado se deve, naturalmente, à forma como as primitivas são implementadas internamente pelas plataformas de virtualização. Por utilizarem códigos proprietários, estes aspectos de mais baixo nível das plataformas utilizadas não puderam ser analisados. Contudo, em alguns casos, foi possível comparar os tempos das operações implementadas com pelos *hypervisors* com os tempos das operações realizadas com auxílio do sistema operacional subjacente, obtendo tempos aproximados de execução. Isso permitiu inferir que as operações testadas nas plataformas selecionadas fazem uso de chamadas de sistema similares àquelas utilizadas pelos sistemas operacionais convencionais.

5.3 Caminhos de Pesquisa para Futuras Investigações

Este trabalho apresentou uma interface de gerenciamento de dispositivos, para manipular os recursos de redes virtuais. Estas interfaces permitem a execução de operações sem que o operador da rede precise conhecer detalhes das implementações internas dos dispositivos. O protocolo SNMP foi escolhido como um ponto inicial para identificação dos requisitos gerais para interfaces de gerenciamento aplicadas à virtualização de redes. Outras tecnologias de gerenciamento de rede, além do SNMP, poderiam ser igualmente utilizadas com este propósito, tais como NETCONF (ENNS, 2006) e Web Services (FERRIS et al., 2002), sendo um tópico de pesquisa relevante, a curto prazo, a análise comparativa entre elas.

Além disso, outras tecnologias de virtualização (ex., OpenVZ, VirtualBox, etc.) e roteamento (ex., Quagga, XORP, etc.) também poderiam ter sido selecionadas. A escolha de diferentes tecnologias tem impacto direto não apenas no desempenho da solução implementada, mas também no custo de implementação. Como exemplo, neste trabalho, o esforço de implementação para o VMWare Server foi bem maior do que para o Xen Server.

O gerenciamento de redes virtuais é um tópico amplo que não se esgota com as interfaces de gerenciamento, englobando uma ampla variedade de aspectos com-

plementares, que precisam ser igualmente investigados e desenvolvidos. Mosharaf K. Chowdhury e Raouf Boutaba (CHOWDHURY; BOUTABA, 2009) levantaram alguns desafios de pesquisa, a longo prazo, que merecem atenção:

- **Interfaceamento -** Neste trabalho foi apresentada uma interface padrão para tornar a programação dos elementos de rede disponível aos SPs. Adicionalmente, todo InP deve fornecer uma interface, seguindo algum padrão, de forma que os SPs possam se comunicar com ele expressando suas necessidades. No gerenciamento entre redes virtuais, com a existência de várias redes virtuais no mesmo substrato, naturalmente podem surgir situações onde usuários/serviços de redes distintas precisam interagir. Dessa forma, interfaces apropriadas entre usuários finais e SPs, bem como entre múltiplos InPs e múltiplos SPs devem ser identificadas e padronizadas;

- **Descoberta de Recursos e Topologias -** A fim de alocar recursos para as requisições de diferentes SPs, os InPs devem ser capazes de determinar a topologia das redes que eles gerenciam bem como o status dos elementos de rede correspondentes (ex., nodos físicos e interconexões entre eles, capacidades restantes em nodos e enlaces, etc.). Do ponto de vista de um SP, uma VN deveria ser capaz de descobrir a presença e topologia de outras VNs coexistentes. Isso possibilitará a comuincação, interação e colaboração entre VNs para fornecer serviços melhores e mais complexos.

- **Controle de Admissão e Policiamento do Uso -** Ao estabelecer uma rede virtual, um SP deve solicitar garantias para alguns atributos de seus nodos e enlaces virtuais. Os InPs devem realizar a contabilidade precisa e implementar algoritmos de controle de admissão e policiamento do uso para garantir a entrega do desempenho solicitado e que as VNs não excedam a alocação de recuros localmente ou globalmente.

- **Gerenciamento de Falhas -** Falhas nos componentes da rede física subjacente podem dar origem a uma série de falhas em cadeia por todas as VNs diretamente instaladas sobre estes componentes. A detecção, propagação e isolação de tais falhas, bem como a proteção e restauração delas são todas questões de pesquisa em aberto que devem ser atacadas.

- **Segurança e Privacidade -** A isolação entre VNs coexistentes pode fornecer apenas um certo nível de segurança e privacidade com o uso de túneis seguros, encriptações, etc. No entanto, as principais ameaças, intrusões e ataques para as redes físicas e virtuais ainda não foram identificados e explorados. Por exemplo, a programação de elementos de rede poderia aumentar a vulnerabilidade se modelos de programação e interfaces seguras não estiverem disponíveis. Todos estes tópicos requerem uma investigação aprofundada.

Somam-se a estes, os seguintes tópicos de estudo, igualmente importantes:

- **Gerenciamento inter-domínios -** Uma rede virtual pode ser composta por roteadores localizados em múltiplos domínios administrativos (ou sistemas autônomos), cada um deles administrados por operadores diferentes. Neste caso, o administrador da rede virtual deve interagir com os diversos administradores das redes físicas que compões a rede virtual. Dois InPs adjacentes

devem ser capazes de instanciar enlaces virtuais inter-domínios para formar redes virtuais (VNs) fim-a-fim. Tal interação não é hoje contemplada em soluções de gerenciamento convencionais;

- **Gerenciamento de desempenho -** Com o advento da virtualização surgem questões de desempenho importantes, já que diversos roteadores virtuais competem pelo poder computacional do roteador físico que os abriga. Conforme mostrado nas avaliações experimentais 1 e 2, a criação excessiva de roteadores virtuais poderia prejudicar o funcionamento do roteador físico associado. Por outro lado, é importante maximizar o número de redes virtuais co-existentes a fim de otimizar a utilização e rentabilidade dos InPs. A alocação otimizada de redes virtuais foi classificada como um problema "NP-Hard"(CHOWDHURY; RAHMAN; BOUTABA, 2009). Porém, o desenvolvimento de algoritmos e heurísticas considerando um conjunto selecionado de parâmetros aplicados à diferentes topologias permanece como um assunto importante de pesquisa.

- **Gerenciamento heterogêneo -** Redes virtuais podem ser compostas por tecnologias de rede variadas, cada uma com suas particularidades de gerenciamento. É importante que estas soluções de gerência individuais possam atuar de forma integrada em um ambiente virtualizado, a fim de prover, operar e manter as diferentes redes;

- **Gerenciamento de qualidade de serviço (QoS) -** Os serviços que compõem uma rede virtual podem possuir diferentes restrições de desemplenho ou exigir uma diferenciação na alocação dos recursos virtuais. A maioria das soluções atuais de gerenciamento não se mostram adequadas para acomodar estas demandas de qualidade de serviço em redes virtuais.

REFERÊNCIAS

AGRAWAL, M.; BAILEY, S. R.; GREENBERG, A.; PASTOR, J.; SEBOS, P.; SESHAN, S.; MERWE, K. van der; YATES, J. RouterFarm: towards a dynamic, manageable network edge. In: SIGCOMM WORKSHOP ON INTERNET NETWORK MANAGEMENT, INM, 2008., Pisa, Italy. **Proceedings...** New York: NY: ACM, 2006. p.5–10.

ANDERSON, T.; PETERSON, L.; SHENKER, S.; TURNER, J. Overcoming the Internet Impasse through Virtualization. **Computer**, Los Alamitos, CA, USA, v.38, n.4, p.34–41, 2005.

ANHALT, F.; KOSLOVSKI, G.; PRIMET, P. V.-B. Specifying and provisioning virtual infrastructures with HIPerNET. **International Journal of Network Management**, [S.l.], v.20, p.129–148, 2010.

AUTOI. **AutoI - Autonomic Internet**. Disponível em: <http://ist-autoi.eu/autoi/index.php >. Acesso em: Fevereiro 2011.

BOUCADAIR, M.; GEORGATSOS, P.; WANG, N.; GRIFFIN, D.; PAVLOU, G.; HOWARTH, M.; ELIZONDO, A. The AGAVE approach for network virtualization: differentiated services delivery. **Annals of Telecommunications**, [S.l.], v.64, p.277–288, 2009. 10.1007/s12243-009-0103-4.

CABO. **CABO (Concurrent Architectures are Better than One)**. Disponível em: <http://www.cs.princeton.edu/ jrex/virtual.html >. Acesso em: Outubro 2010.

CHOWDHURY, N. M. K.; BOUTABA, R. IEEE Communications magazine: network virtualization: state of the art and research challenges. **Network and Service Management**, New York, NY, USA, v.47, n.7, p.20–26, July 2009.

CHOWDHURY, N.; RAHMAN, M.; BOUTABA, R. Virtual Network Embedding with Coordinated Node and Link Mapping. In: INFOCOM 2009, Rio de Janeiro, RJ, Brazil. **Proceedings...** IEEE, 2009. p.783–791.

CISCO. **Logical Routers Commands on Cisco IOS XR Software**. Disponível em: <http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html >. Acesso em: Junho 2010.

CISCOXR. **Cisco XR 12000 Series Router**. Disponível em: <http://www.cisco.com/en/US/products/ps6342/index.html >. Acesso em: Dezembro 2010.

COMER, D.; MARTYNOV, M. Building experimental virtual routers with network processors. In: INTERNATIONAL CONFERENCE ON TESTBEDS AND RESEARCH INFRASTRUCTURES FOR THE DEVELOPMENT OF NETWORKS AND COMMUNITIES, TRIDENTCOM, 2006., Barcelona, Spain. **Proceedings...** IEEE, 2006. p.9–230.

DANIELE, M.; WIJNEN, B.; ELLISON, E. M.; FRANCISCO, E. D. **Agent Extensibility (AgentX) Protocol - Version 1**. [S.l.]: Internet Engineering Task Force, Network Working Group, 2000.

ENNS, E. R. **NETCONF Configuration Protocol**. [S.l.]: Internet Engineering Task Force, Network Working Group, 2006.

FERRIS, C.; FALLSIDE, D.; MILLER, E.; HAAS, H.; MARSH, J. **Web Services**. Disponível em: <http://www.w3.org/2002/ws/ >. Acesso em: Janeiro 2011.

GALáN, F.; FERNáNDEZ, D.; FUERTES, W.; GóMEZ, M.; VERGARA, J. L. de. Scenario-based virtual network infrastructure management in research and educational testbeds with VNUML. **Annals of Telecommunications**, [S.l.], v.64, p.305–323, 2009. 10.1007/s12243-009-0104-3.

HANDLEY, M. **XORP eXtensible Open Router Platform**. Disponível em: <http://www.xorp.org/ >. Acesso em: Junho 2010.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for Describing SNMP Management Frameworks**. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

HOU, T.-C.; CHAN, M.-C.; YU, C.-T. Using tunneling techniques to realize virtual routers. **Annals of Telecommunications**, [S.l.], v.64, p.325–338, 2009. 10.1007/s12243-009-0105-2.

IPERF. **Iperf 2.0.4**. Disponível em: <http://sourceforge.net/projects/iperf >. Acesso em: Junho 2010.

JUNIPER. **Consolidating Service Provider Points of Presence**. Sunnyvale, CA, USA: Juniper Networks, Inc., 2010. (2000343-001-EN).

JUNOS. **Logical Routers**. Disponível em: <http://www.juniper.net/techpubs /software/junos/junos85/feature-guide-85/id-11139212.html >. Acesso em: Junho 2010.

KNIGHT, P.; OULD-BRAHIM, H.; WRIGHT, G.; GLEESON, B.; SLOANE, T.; BUBENIK, R.; SARGOR, C.; NEGUSSE, I.; YU, J. J.; BACH, R.; YOUNG, A.; FANG, L.; WEBER, D. C. **Network based IP VPN Architecture using Virtual Routers**. [S.l.]: Internet Engineering Task Force, Provider Provisioned VPN Working Group, 2003.

KOLON, M. **Intelligent Logical Router Service**. Sunnyvale, CA, USA: Juniper Networks, Inc., 2004. (200097-001).

KVM. **KVM**: kernel-based virtual machine. Disponível em: <http://www.linux-kvm.org >. Acesso em: Dezembro 2010.

L3VS. **Layer 3 Virtual Switching**. Santa Clara, CA, USA: Extreme Networks, Inc., 2007. (1185_01).

LEMAY, M. **An Introduction to IaaS Framework - Reference Architecture Framework for Providing IaaS**. [S.l.]: Inocybe Technologies Inc., 2008.

LIBES, D. **Expect TCL/Tk Extensions**. Disponível em: <http://sourceforge.net/projects/expect/ >. Acesso em: Junho 2010.

MAIER, S.; HERRSCHER, D.; ROTHERMEL, K. Experiences with node virtualization for scalable network emulation. **Comput. Commun.**, Newton, MA, USA, v.30, n.5, p.943–956, 2007.

MCCLOGHRIE, K.; PERKINS, D.; SCHOENWAELDER, J. **Structure of Management Information Version 2 (SMIv2)**. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

NET-SNMP. **Net-SNMP Tools**. Disponível em: <http://www.net-snmp.org/ >. Acesso em: Junho 2010.

NG, W.; JUN, D.; CHOW, H.; BOUTABA, R.; LEON-GARCIA, A. MIBlets: a practical approach to virtual network management. In: SIXTH IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 1999., Boston, MA , USA. **Proceedings...** IEEE, 2002. p.201–215. (Distributed Management for the Networked Millennium).

NICK FEAMSTER, L. G.; REXFORD, J. **Cabo**: concurrent architectures are better than one. [S.l.]: Cisco Inc., 2008.

OPENVSWITCH. **Open vSwitch**: an open virtual switch. Disponível em: <http://openvswitch.org/ >. Acesso em: Junho 2010.

PAPADIMITRIOU, P.; MAENNEL, O.; GREENHALGH, A.; FELDMANN, A.; MATHY, L. Implementing Network Virtualization for a Future Internet. In: ITC SPECIALIST SEMINAR ON NETWORK VIRTUALIZATION, 2009., Hoi An, Viet Nam. **Proceedings...** IEEE, 2009.

PARALLELS. **Parallels Desktop 4 for Windows Linux**. Disponível em: <www.parallels.com/products/workstation/ >. Acesso em: Dezembro 2010.

QUAGGA. **Quagga Routing Suite**. Disponível em: <http://www.quagga.net/ >. Acesso em: Junho 2010.

RIXNER, S. Network Virtualization: breaking the performance barrier. **Queue Virtualization Magazine**, New York, NY, USA, v.6, n.1, p.36–ff, Feb. 2008.

RODRíGUEZ-HARO, F.; FREITAG, F.; NAVARRO, L. Enhancing virtual environments with QoS aware resource management. **Annals of Telecommunications**, Paris, France, v.64, n.5, p.289–303, Apr. 2009.

ROSTAMI, A.; SARGENT, E. H. An Optical Integrated system for Implementation of Optical Cross-connect, Beam Splitter, Mux/demux and Combiner. **Journal of Computer Science and Network Security**, [S.l.], v.6, p.43–49, 2006.

SCHONWALDER, J.; PRAS, A.; MARTIN-FLATIN, J.-P. On the future of Internet management technologies. **IEEE Communications Magazine**, [S.l.], v.41, n.10, p.90–97, Oct. 2003.

SINET3. **Science Information Network 3 (SINET3)**. Disponível em: <http://www.sinet.ad.jp/topology >. Acesso em: Janeiro 2011.

SPYROPOULOS, T.; FDIDA, S.; KIRKPATRICK, S. Future internet: fundamentals and measurement. **SIGCOMM Comput. Commun. Rev.**, New York, NY, USA, v.37, p.101–106, March 2007.

STELZER, E.; HANCOCK, S.; SCHLIESSER, B.; LARIA, J. **Virtual Router Management Information Base Using SMIv2**. [S.l.]: Internet Engineering Task Force, Provider Provisioned VPN Working Group, 2003.

TAYLOR, D.; TURNER, J. **Towards a diversified Internet**. [S.l.]: Washington University in Saint Louis, 2004.

TOUCH, J. Dynamic Internet Overlay Deployment and Management Using the X-Bone. **Computer Networks**, [S.l.], v.36, n.2–3, p.117–135, jul 2001.

TOUCH, J. D.; WANG, Y.-S.; EGGERT, L.; FINN, G. G. A Virtual Internet Architecture. **ACM SIGCOMM Workshop on Future Directions in Network Architecture**, Karlsruhe, Germany, mar 2003.

TSUGAWA, M.; FORTES, J. A. B. Characterizing user-level network virtualization: performance, overheads and limits. **International Journal of Network Management**, [S.l.], v.20, p.149–166, 2010.

TURNER, J.; MCKEOWN, N. Can overlay hosting services make ip ossification irrelevant? In: WORKSHOP ON PROGRAMMABLE ROUTERS FOR THE EXTENSIBLE SERVICES OF TOMORROW, PRESTO, 2007., Seattle, WA, USA. **Proceedings...** [S.l.: s.n.], 2007.

TXMATRIX. **Juniper TX Matrix Plus Routers**. Disponível em: <http://www.juniper.net/products_and_services/t_series_core_platforms/index.html >. Acesso em: Dezembro 2010.

URUSHIDANI, S.; FUKUDA, K.; JI, Y.; ABE, S.; KOIBUCHI, M.; NAKAMURA, M.; YAMADA, S.; SHIMIZU, K.; HAYASHI, R.; INOUE, I.; SHIOMOTO, K. Resource Allocation and Provision for Bandwidth/Networks on Demand in SINET3. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS, NOMS, 2008., Salvador, BA, Brazil. **Proceedings...** IEEE, 2008. p.212–218.

VIRTUALBOX, O. **Oracle VirtualBox**. Disponível em: <http://www.virtualbox.org/ >. Acesso em: Dezembro 2010.

VIRTUALPC. **Windows Virtual PC**. Disponível em: <http://www.microsoft.com/windows/virtual-pc/ >. Acesso em: Dezembro 2010.

VMSERVER. **VMware Server**. Disponível em: <http://www.vmware.com/products/server/ >. Acesso em: Dezembro 2010.

VMWARE. **VMware Virtualization Software for Desktops, Servers Virtual Machines**. Disponível em: <http://www.vmware.com >. Acesso em: Junho 2010.

VMWAREESX. **VMware ESXi/ESX**. Disponível em: <http://www.vmware.com/products/vsphere/esxi-and-esx/ >. Acesso em: Dezembro 2010.

VMWORKSTATION. **VMware Workstation Desktop Virtualization**. Disponível em: <https://www.vmware.com/products/workstation/ >. Acesso em: Dezembro 2010.

VYATTA. **Vyatta Open Networking**. Disponível em: <http://www.vyatta.com >. Acesso em: Junho 2010.

VYATTA-REPLACEMENT. **Vyatta Replacement Guide - Cisco**. Disponível em: <http://www.vyatta.com/downloads/other/Vyatta_Cisco_Replacement_Guide.pdf >. Acesso em: Junho 2010.

WANG, Y.; KELLER, E.; BISKEBORN, B.; MERWE, J. van der; REXFORD, J. Virtual routers on the move: live router migration as a network-management primitive. In: ACM SIGCOMM 2008 CONFERENCE ON DATA COMMUNICATION, 2008., Seattle, WA, USA. **Proceedings...** New York: NY: ACM, 2008. p.231–242.

XEN. **XenServer Hypervisor**. Disponível em: <http://www.xen.org/ >. Acesso em: Junho 2010.

APÊNDICE A DETALHES DA SOLUÇÃO PROPOSTA

Neste Apêndice são apresentados alguns dos códigos fonte utilizados na implementação da interface e do agente proposto.

A interface foi desenvolvida a partir da VIRTUAL-ROUTER-MIB (STELZER et al., 2003), com modificações destacadamente presentes no grupo de configuração e associação de interfaces de rede ("vrIfConfig"). O Apêndice A.1 apresenta a descrição da MIB implementada.

Listing A.1: Código Fonte para MIB Virtual Router

```
 1 --
 2 -- VIRTUAL-ROUTER-MIB
 3 --
 4
 5 VIRTUAL-ROUTER-MIB DEFINITIONS ::= BEGIN
 6
 7     IMPORTS
 8         InterfaceIndex
 9             FROM IF-MIB
10         InetAddressType, InetAddress
11             FROM INET-ADDRESS-MIB
12         OBJECT-GROUP, MODULE-COMPLIANCE, NOTIFICATION-GROUP
13             FROM SNMPv2-CONF
14         experimental, Unsigned32, OBJECT-TYPE,
15         MODULE-IDENTITY, TimeTicks, NOTIFICATION-TYPE
16             FROM SNMPv2-SMI
17         TruthValue, DisplayString, RowStatus, PhysAddress, TEXTUAL-CONVENTION
18             FROM SNMPv2-TC;
19
20     virtualRouterMIB MODULE-IDENTITY
21         LAST-UPDATED "200301311200Z"
22         ORGANIZATION
23             "IETF PPVPN WG"
24         CONTACT-INFO
25             "
26             Elwin Stelzer Eliazer
27             Corona Networks, Inc.
28             630 Alder Drive
29             Milpitas, CA 95035
30             USA
31             Phone: +1-408-519-3832
32             Email: elwinietf@yahoo.com
33
34             Samuel Hancock
35             Corona Networks, Inc.
36             630 Alder Drive
37             Milpitas, CA 95035
38             USA
39             Phone: +1-408-519-3800 Ext 421
40             Email: hancoc_s@yahoo.com
41
42             Benson Schliesser
43             SAVVIS Communications
44             1 Savvis Parkway
45             Town and Country, MO 63017
46             USA
47             Phone: +1-314-628-7036
48             Email: bensons@savvis.net
49
50             Joseph Laria
51             Level Stream Research
52             Wilmington, MA  01887
53             USA
54             Phone: +1-978-884-3537
55             Emain: jlaria@levelstream.com
56             "
57         DESCRIPTION
58             "The MIB is the definition of the managed
59             objects for the Virtual Router."
60         REVISION "200306011200Z"
```

```
        DESCRIPTION
            "VR-MIB Draft of the IETF PPVPN WG"
        ::= { experimental 59 } -- To be assigned

--
-- Textual conventions
--

    VrId ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "Virtual Router Identifier.
             VRID 0 is reserved for the Administrative VR
             and cannot be used to create VR's.
            "
        SYNTAX Unsigned32

    VpnIdentifier ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "RFC2685:  The global VPN Identifier format is:
            3 octet VPN authority Organizationally Unique Identifier
            followed by
            4 octet VPN index identifying VPN according to OUI"
        SYNTAX OCTET STRING(SIZE (0..7))

--
-- Node definitions
--

    vrMIBObjects OBJECT IDENTIFIER ::= { virtualRouterMIB 1 }

    vrConfig OBJECT IDENTIFIER ::= { vrMIBObjects 1 }

    vrConfigScalars OBJECT IDENTIFIER ::= { vrConfig 1 }

    vrConfigNextAvailableVrId OBJECT-TYPE
        SYNTAX VrId
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The next available Virtual Router Id (index).
            This object provides a hint for the vrID value
            to use when administratively creating a new
            vrConfigEntry.

            A GET of this object returns the next available vrId
            value to be used to create an entry in the associated
            vrConfigTable; or zero, if no valid vrId
            value is available. A value of zero(0) indicates that
            it is not possible to create a new vrConfigEntry
            This object also returns a value of zero when it is the
            lexicographic successor of a varbind presented in an
            SNMP GETNEXT or GETBULK request, for which circumstance
            it is assumed that ifIndex allocation is unintended.

            Successive GETs will typically return different
            values, thus avoiding collisions among cooperating
            management clients seeking to create table entries
            simultaneously.

            Unless specified otherwise by its MAX-ACCESS and
            DESCRIPTION clauses, an object of this type is read-only,
            and a SET of such an object returns a notWritable error."
        ::= { vrConfigScalars 1 }

    vrConfigTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VrConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table is for creating the new Virtual Routers."
        ::= { vrConfig 2 }

    vrConfigEntry OBJECT-TYPE
        SYNTAX VrConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The entries in this table can be added/deleted
            using the vrRowStatus."
        INDEX { vrId }
        ::= { vrConfigTable 1 }

    VrConfigEntry ::=
        SEQUENCE {
            vrId
                VrId,
            vrRowStatus
                RowStatus,
            vrName
                DisplayString,
            vrContextName
                DisplayString,
            vrTrapEnable
                TruthValue,
            vrMaxRoutes
                Unsigned32,
            vrAdminStatus
```

```
160                 INTEGER,
161             vrVpnId
162                 VpnIdentifier,
163             vrRpTrigger
164                 Unsigned32
165          }
166
167     vrId OBJECT-TYPE
168         SYNTAX VrId
169         MAX-ACCESS not-accessible
170         STATUS current
171         DESCRIPTION
172             "The unique id of this virtual router instance. A Virtual
173              Router cannot not be created with vrId = 0.
174              VRID 0 is reserved for the Administrative VR.
175             "
176     ::= { vrConfigEntry 1 }
177
178     vrRowStatus OBJECT-TYPE
179         SYNTAX RowStatus
180         MAX-ACCESS read-create
181         STATUS current
182         DESCRIPTION
183             "The status column has three defined values:
184
185             - 'active', which indicates that the conceptual row is
186             available for use by the managed device;
187
188             - 'createAndGo', which is supplied by a management
189             station wishing to create a new instance of a
190             conceptual row and to have its status automatically set
191             to active, making it available for use by the managed
192             device;
193
194             - 'destroy', which is supplied by a management station
195             wishing to delete all of the instances associated with
196             an existing conceptual row."
197     ::= { vrConfigEntry 2 }
198
199     vrName OBJECT-TYPE
200         SYNTAX DisplayString
201         MAX-ACCESS read-create
202         STATUS current
203         DESCRIPTION
204             "The Name of the Virtual Router..
205              "
206         ::= { vrConfigEntry 3 }
207
208     vrContextName OBJECT-TYPE
209         SYNTAX DisplayString
210         MAX-ACCESS read-create
211         STATUS current
212         DESCRIPTION
213             "The SNMPv2 Community String or SNMPv3 contextName
214             denotes the VR 'context' and is used to logically
215             separate the MIB management.
216             RFC2571 and RFC2737 describe this approach."
217         ::= { vrConfigEntry 4 }
218
219     vrTrapEnable OBJECT-TYPE
220         SYNTAX TruthValue
221         MAX-ACCESS read-create
222         STATUS current
223         DESCRIPTION
224             "This objects is used to enable the generation
225             of the VrUp and VrDown traps.
226                 true(1)     - VR Traps Enabled
227                 false(2)    - VR Traps Disabled"
228         DEFVAL { true }
229         ::= { vrConfigEntry 5 }
230
231     vrMaxRoutes OBJECT-TYPE
232         SYNTAX Unsigned32
233         MAX-ACCESS read-create
234         STATUS current
235         DESCRIPTION
236             "This object specifies the maximum number of routes that
237             this VR can support. The default value is 4 Gig (meaning
238             unlimited)."
239         DEFVAL { 4294967295 }
240         ::= { vrConfigEntry 6 }
241
242     vrAdminStatus OBJECT-TYPE
243         SYNTAX  INTEGER {
244                  up(1),
245                  down(2),
246                  testing(3),
247                  unknown(4)
248                 }
249         MAX-ACCESS read-create
250         STATUS current
251         DESCRIPTION
252             "The administrative state of the Virtual Router."
253         DEFVAL { down }
254         ::= { vrConfigEntry 7 }
255
256     vrVpnId OBJECT-TYPE
257         SYNTAX  VpnIdentifier
258         MAX-ACCESS read-create
```

```
        STATUS current
        DESCRIPTION
            "The Virtual Private Network Identifier of the Virtual
             Router."
        ::= { vrConfigEntry 8 }

    vrRpTrigger OBJECT-TYPE
        SYNTAX  Unsigned32
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The Routing Protocol Triggers on the Virtual Router.
            This can be used to initiate or shutdown routing protocols
            on a VR.
            The 32 bits are divided into:
                16 bits of RP bitmap,
                15 bits reserved (0), and 1 bit of action-code.


                         0 1 2 3 4 5 6 7
                        +-+-+-+-+-+-+-+-+
                        |RP bitmap (MSB)|
                        +-+-+-+-+-+-+-+-+
                        |RP bitmap (LSB)|
                        +-+-+-+-+-+-+-+-+
                        |   Reserved    |
                        +-+-+-+-+-+-+-+-+
                        |*|  Reserved   |
                        +-+-+-+-+-+-+-+-+    *=action-code


            The RP bitmap specify the RP that is to be initiated or
            shutdown. Multiple RPs can be acted on simultaneously.
            Also, individual RPs can be brought up in steps, which
            should not affect the RPs that were running.
            Action-code specify what needs to be done for the RPs
            in the RP bitmap.  The actions are: initiate or shutdown.

            The running status of the RP shall be available in the
            VR stats table's vrRpStatus, which has a similar
            format, but represent the status."
        ::= { vrConfigEntry 9 }

    vrStat OBJECT IDENTIFIER ::= { vrMIBObjects 2 }

    vrStatScalars OBJECT IDENTIFIER ::= { vrStat 1 }

    vrConfiguredVRs OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of VRs configured on this network element."
        ::= { vrStatScalars 1 }

    vrActiveVRs OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of VRs that are active on the network element.
            These are VRs for which the
            vrStatOperationalStatus  = up(1)"
        ::= { vrStatScalars 2 }

    vrStatTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VrStatEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table contains statistics for the Virtual Router."
        ::= { vrStat 2 }

    vrStatEntry OBJECT-TYPE
        SYNTAX VrStatEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Entries in this table a per vrId."
        INDEX { vrId }
        ::= { vrStatTable 1 }

    VrStatEntry ::=
        SEQUENCE {
            vrStatRouteEntries
                Unsigned32,
            vrStatFIBEntries
                Unsigned32,
            vrStatUpTime
                TimeTicks,
            vrOperStatus
                INTEGER,
            vrRpStatus
                Unsigned32,
            vrRouterAddressType
                InetAddressType,
            vrRouterAddress
                InetAddress
         }
```

```
358
359     vrStatRouteEntries OBJECT-TYPE
360         SYNTAX Unsigned32
361         MAX-ACCESS read-only
362         STATUS current
363         DESCRIPTION
364             "Total number of routes for this VR."
365         ::= { vrStatEntry 1 }
366
367     vrStatFIBEntries OBJECT-TYPE
368         SYNTAX Unsigned32
369         MAX-ACCESS read-only
370         STATUS current
371         DESCRIPTION
372             "Total number of FIB Entries for this VR."
373         ::= { vrStatEntry 2 }
374
375     vrStatUpTime OBJECT-TYPE
376         SYNTAX TimeTicks
377         MAX-ACCESS read-only
378         STATUS current
379         DESCRIPTION
380             "The time in (in hundredths of a second) since
381             this VR entry has been operational."
382         ::= { vrStatEntry 3 }
383
384     vrOperStatus OBJECT-TYPE
385         SYNTAX  INTEGER {
386                  up(1),
387                  down(2),
388                  unknown(3)
389                 }
390         MAX-ACCESS read-only
391         STATUS current
392         DESCRIPTION
393             "The operational state of the Virtual Router."
394         ::= { vrStatEntry 4 }
395
396     vrRpStatus OBJECT-TYPE
397         SYNTAX  Unsigned32
398         MAX-ACCESS read-only
399         STATUS current
400         DESCRIPTION
401             "List of Routing Protocols on this VR."
402         ::= { vrStatEntry 5 }
403
404     vrRouterAddressType OBJECT-TYPE
405         SYNTAX  InetAddressType
406         MAX-ACCESS read-only
407         STATUS current
408         DESCRIPTION
409             "Router Address Type of this VR."
410         ::= { vrStatEntry 6 }
411
412     vrRouterAddress OBJECT-TYPE
413         SYNTAX  InetAddress
414         MAX-ACCESS read-only
415         STATUS current
416         DESCRIPTION
417             "Router Address of this VR.  It is derived from one of the
418             interfaces. If loopback interface is present, the loopback
419             interface address can be used. However, loopback interface
420             is optional."
421         ::= { vrStatEntry 7 }
422
423     vrIfConfig OBJECT IDENTIFIER ::= { vrMIBObjects 3 }
424
425     vrIfConfigScalars OBJECT IDENTIFIER ::= { vrIfConfig 1 }
426
427     vrIfConfigTable OBJECT-TYPE
428         SYNTAX SEQUENCE OF VrIfConfigEntry
429         MAX-ACCESS not-accessible
430         STATUS current
431         DESCRIPTION
432             "This table is for configuring VR Interfaces."
433         ::= { vrIfConfig 2 }
434
435     vrIfConfigEntry OBJECT-TYPE
436         SYNTAX VrIfConfigEntry
437         MAX-ACCESS not-accessible
438         STATUS current
439         DESCRIPTION
440             "Entries in this table correspond to the entries in
441             the ifTable that apply to the Virtual Router."
442         INDEX { vrId,
443                 vrIfId }
444         ::= { vrIfConfigTable 1 }
445
446     VrIfConfigEntry ::=
447         SEQUENCE {
448             vrIfId
449                InterfaceIndex,
450             vrIfConfigRowStatus
451                RowStatus,
452             vrIfName
453                DisplayString,
454             vrIfMac
455                PhysAddress
456          }
```

```
457
458     vrIfId OBJECT-TYPE
459         SYNTAX InterfaceIndex
460         MAX-ACCESS not-accessible
461         STATUS current
462         DESCRIPTION
463             "Virtual Router Interface Index."
464         ::= { vrIfConfigEntry 1 }
465
466      vrIfConfigRowStatus  OBJECT-TYPE
467         SYNTAX RowStatus
468         MAX-ACCESS read-create
469         STATUS current
470         DESCRIPTION
471             " This object is used to create, delete or
472             modify a row in this table."
473         ::= { vrIfConfigEntry 2 }
474
475     vrIfName OBJECT-TYPE
476         SYNTAX DisplayString
477         MAX-ACCESS read-create
478         STATUS current
479         DESCRIPTION
480             "The logical name for an interface/type."
481         ::= { vrIfConfigEntry 3 }
482
483     vrIfMac OBJECT-TYPE
484         SYNTAX PhysAddress
485         MAX-ACCESS read-create
486         STATUS current
487         DESCRIPTION
488             "The generated MAC address for the
489             virtual interface."
490         ::= { vrIfConfigEntry 4 }
491
492 -- ****************************************************************
493 -- Module Traps/Notifications
494 -- ****************************************************************
495
496     vrNotificationsPrefix OBJECT IDENTIFIER ::= { vrMIBObjects 4 }
497
498     vrNotifications OBJECT IDENTIFIER ::= { vrNotificationsPrefix 0 }
499
500     vrUp NOTIFICATION-TYPE
501         OBJECTS { vrRowStatus }
502         STATUS current
503         DESCRIPTION
504             "This notification is generated when the specified
505             VR is about to initialized or change the status from
506             down to up."
507         ::= { vrNotifications 1 }
508
509     vrDown NOTIFICATION-TYPE
510         OBJECTS { vrRowStatus }
511         STATUS current
512         DESCRIPTION
513             "This notification is generated when the specified
514             VR is about to go down."
515         ::= { vrNotifications 2 }
516
517     vrMaxRoutesExceeded NOTIFICATION-TYPE
518         OBJECTS { vrRowStatus, vrMaxRoutes, vrStatRouteEntries }
519         STATUS current
520         DESCRIPTION
521             "This notification is generated when the specified VR has
522             exceeded the maximum number of routes specified"
523         ::= { vrNotifications 3 }
524
525
526 -- ****************************************************************
527 -- Module Compliance/Conformance Statements
528 -- ****************************************************************
529
530     vrConformance OBJECT IDENTIFIER ::= { virtualRouterMIB 2 }
531
532     vrCompliances OBJECT IDENTIFIER ::= { vrConformance 1 }
533
534     vrMIBCompliance MODULE-COMPLIANCE
535         STATUS current
536         DESCRIPTION
537             "The compliance statement for entities that implement the
538             VIRTUAL-ROUTER-MIB.  Implementation of this MIB
539             is strongly recommended for any platform targeted for a
540             carrier-class environment."
541         MODULE -- this module
542             MANDATORY-GROUPS { vrConfigGroup, vrStatGroup,
543                                vrIfGroup, vrNotificationGroup }
544         ::= { vrCompliances 1 }
545
546     vrGroups OBJECT IDENTIFIER ::= { vrConformance 2 }
547
548     vrConfigGroup OBJECT-GROUP
549         OBJECTS { vrRowStatus, vrName,
550                   vrContextName, vrTrapEnable,
551                   vrMaxRoutes, vrAdminStatus,
552                   vrVpnId, vrRpTrigger,
553                   vrConfigNextAvailableVrId }
554             STATUS current
555             DESCRIPTION
```

```
556                 "A collection of attributes that support provisioning
557                 of a virtual router."
558             ::= { vrGroups 1 }
559
560     vrStatGroup OBJECT-GROUP
561         OBJECTS { vrConfiguredVRs, vrActiveVRs,
562                   vrStatRouteEntries, vrStatFIBEntries, vrStatUpTime,
563                   vrOperStatus, vrRpStatus, vrRouterAddress,
564                   vrRouterAddressType }
565         STATUS current
566         DESCRIPTION
567             "A collection of attributes that contain stats about the
568             virtual router."
569         ::= { vrGroups 2 }
570
571     vrIfGroup OBJECT-GROUP
572         OBJECTS { vrIfConfigRowStatus  }
573         STATUS current
574         DESCRIPTION
575             "A collection of attributes that support provisioning of a
576             virtual router interfaces."
577         ::= { vrGroups 3 }
578
579     vrNotificationGroup NOTIFICATION-GROUP
580         NOTIFICATIONS { vrUp, vrDown, vrMaxRoutesExceeded }
581         STATUS current
582         DESCRIPTION
583             "A collection of traps that are supported by the VR."
584         ::= { vrGroups 4 }
585
586 END
```

O agente foi implementado com o NET-SNMP 5.5, utilizando os arquivos virtualRouterMIB.h, virtualRouterMIB.c, vrIfConfigTable.h e vrIfConfigTable.c localizados no diretório . . . /net-snmp-5.5/agent/mibgroup/virtual-router-mib.

O código fonte do módulo do agente SNMP implementado para VIRTUAL-ROUTER-MIB encontra-se disponível, sob condições de uso da licença GNU GPL (General Public License), nos endereços a seguir:

Listing A.2: URL para download do código fonte

```
1 Versão VMware Server - http://www.inf.ufrgs.br/~ffdaitx/source_code-vmware.tar.gz         (link principal)
2                       - http://sites.google.com/site/fabiodaitx/source_code-vmware.tar.gz  (link alternativo)
3
4 Versão Xen Server     - http://www.inf.ufrgs.br/~ffdaitx/source_code-xen.tar.gz            (link principal)
5                       - http://sites.google.com/site/fabiodaitx/source_code-xen.tar.gz     (link alternativo)
```

Por questões de organização e espaço o código fonte não foi impresso neste trabalho. A metodologia utilizada para desenvolver o protótipo foi baseada no comando "mib2c" do pacote NET-SNMP 5.5 e seguiu a sequênicia de passos apresentada a seguir.

Listing A.3: Metodologia de design

```
1  Geracao do codigo com o mib2c
2
3  # Certifica-se de que a MIB esta instalada (copiada) junto ao net-snmp para que este consiga le-la
4  root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent/mibgroup# ls -la /usr/local/share/snmp/mibs/ | grep
       VIRTUAL-ROUTER-MIB.txt
5  -rw-r--r-- 1 root root  17762 2010-03-03 01:25 VIRTUAL-ROUTER-MIB.txt
6
7  # Adiciona nome da mib a variavel de ambiente do net-snmp
8  export MIBS+=VIRTUAL-ROUTER-MIB
9
10 # Gera codigo pra toda mib (menos vrIfConfigTable; arquivos de saida ficam no diretorio atual onde e executado o
       comando (cuidado para nao tentar sobreescrever arquivos existentes)
11 mib2c virtualRouterMIB
12
13 # Gera codigo apenas para a tabela de vrIfConfig definida dentro da mib VIRTUAL-ROUTER-MIB
14 # A tabela vai se comportar como um modulo independente do agente, o que facilita o desenvolvimento
15 mib2c vrIfConfigTable
16
17 # Todos arquivos do modulo ficam dentro de um mesmo diretorio
18 root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent/mibgroup/virtual-router-mib# ls -la
19 total 212
20 drwxr-sr-x  3 fabio fabio  4096 2010-05-05 11:06 .
21 drwxr-sr-x 36 fabio fabio  4096 2010-05-05 10:34 ..
22 drwxr-sr-x  2 root  fabio  4096 2010-05-05 11:06 .libs
23 -rwxr-xr-x  1 fabio fabio 41930 2010-05-05 10:49 virtualRouterMIB.c
24 -rw-r--r--  1 fabio fabio 41928 2010-05-05 10:44 virtualRouterMIB.c~
25 -rwxr-xr-x  1 fabio fabio  1710 2010-05-05 09:40 virtualRouterMIB.h
26 -rw-r--r--  1 fabio fabio  1846 2010-05-05 09:40 virtualRouterMIB.h~
27 -rw-r--r--  1 root  fabio   321 2010-05-05 10:49 virtualRouterMIB.lo
28 -rw-r--r--  1 root  fabio 36768 2010-05-05 10:49 virtualRouterMIB.o
```

```
-rwxrwxrwx  1 root  fabio 12786 2010-05-05 11:05 vrIfConfigTable.c
-rw-r--r--  1 fabio fabio 12784 2010-05-05 11:05 vrIfConfigTable.c~
-rwxrwxrwx  1 root  fabio   626 2010-05-05 09:31 vrIfConfigTable.h
-rw-r--r--  1 root  fabio   318 2010-05-05 11:06 vrIfConfigTable.lo
-rw-r--r--  1 root  fabio 22364 2010-05-05 11:06 vrIfConfigTable.o


------------------------------------------------------------------------

Primeira vez, no diretorio do net-snmp:

# verifica diretorio atual (deve estar no diretorio base do netsnmp)
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# pwd
/home/fabio/eclipse/workspace/net-snmp-5.5

# configura para usar modulos que devem ser salvos em agent/mibgroup/<folder-do-modulo>/<arquivos do modulo (
      submodulos)>
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# ./configure --with-mib-modules="virtual-router-mib/
      virtualRouterMIB virtual-router-mib/vrIfConfigTable"

# compila
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# make

# instala
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# make install

------------------------------------------------------------------------

Proximas vezes, nos diretorios agent/mibgroup e agent/:

# economiza muito tempo de compilacao
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# cd agent/mibgroup/
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent/mibgroup# make

# economiza um pouco de tempo de instalacao
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent/mibgroup# cd ..
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent# make install


------------------------------------------------------------------------

Teste de execucao:

# Inicializa agente (dar um netstat antes e matar processo snmpd caso este ja esteja iniciado)
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5/agent/mibgroup# snmpd -c /home/fabio/Desktop/snmpd.conf -Lf /
      home/fabio/Desktop/snmpd.log -DvirtualRouterMIB

# Verifica log de inicializacao
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# cat /home/fabio/Desktop/snmpd.log
registered debug token virtualRouterMIB, 1
registered debug token vrIfConfigTable, 1
virtualRouterMIB: Initializing
virtualRouterMIB:init: initializing table vrConfigTable
virtualRouterMIB:init: initializing table vrStatTable
vrIfConfigTable:init: initializing table vrIfConfigTable
NET-SNMP version 5.5

# Verifica processo/porta do net-snmp
root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5# netstat -ulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 192.168.180.1:137       0.0.0.0:*                           5519/nmbd
udp        0      0 192.168.115.1:137       0.0.0.0:*                           5519/nmbd
udp        0      0 0.0.0.0:137             0.0.0.0:*                           5519/nmbd
udp        0      0 192.168.180.1:138       0.0.0.0:*                           5519/nmbd
udp        0      0 192.168.115.1:138       0.0.0.0:*                           5519/nmbd
udp        0      0 0.0.0.0:138             0.0.0.0:*                           5519/nmbd
udp        0      0 0.0.0.0:48282           0.0.0.0:*                           8368/mgmibbpe
udp        0      0 0.0.0.0:161             0.0.0.0:*                           3990/snmpd
udp        0      0 0.0.0.0:162             0.0.0.0:*                           5806/mgtrapd
udp        0      0 0.0.0.0:36901           0.0.0.0:*                           5250/avahi-daemon:
udp        0      0 192.168.115.1:53        0.0.0.0:*                           5188/named
udp        0      0 192.168.180.1:53        0.0.0.0:*                           5188/named
udp        0      0 127.0.0.1:53            0.0.0.0:*                           5188/named
udp        0      0 0.0.0.0:22222           0.0.0.0:*                           5220/openvpn
udp        0      0 0.0.0.0:37845           0.0.0.0:*                           8368/mgmibbpe
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           5250/avahi-daemon:
udp        0      0 0.0.0.0:54125           0.0.0.0:*                           5188/named
udp        0      0 0.0.0.0:46584           0.0.0.0:*                           5806/mgtrapd
udp6       0      0 :::57502                :::*                                5806/mgtrapd
udp6       0      0 :::41634                :::*                                5188/named
udp6       0      0 :::47523                :::*                                8368/mgmibbpe
udp6       0      0 :::53                   :::*                                5188/named
udp6       0      0 :::40771                :::*                                8368/mgmibbpe
```

A fim de facilitar a recompilação, após quaisquer modificações no código fonte, foi utilizado o script bash “recompila”.

Listing A.4: Script para recompilar o código

```
#!/bin/bash

# a ser rodado no diretorio raiz do net-snmp como root (e.x: root@fabio-laptop:~/eclipse/workspace/net-snmp-5.5#)

proc=$(netstat -ulpn | grep snmpd | awk '{print $6}' | cut -d '/' -f1)
```

```
kill $proc
cd agent/mibgroup
make
cd ..
make install
snmpd -c /home/ubuntu/Desktop/snmpd.conf -Lf /home/ubuntu/Desktop/snmpd.log -DvirtualRouterMIB,vrIfConfigTable
netstat -ulpn
cat /home/ubuntu/Desktop/snmpd.log
```

O código dos scripts implementados para acessar as informações da VIRTUAL-ROUTER-MIB nas plataformas Citrix Xen Server e VMware Server encontra-se disponível, sob condições de uso da licença GNU GPL (General Public License), nos endereços a seguir:

Listing A.5: URL para download do código dos scripts

```
Versão VMware Server - http://www.inf.ufrgs.br/~ffdaitx/scripts-vmware.tar.gz         (link principal)
                     - http://sites.google.com/site/fabiodaitx/scripts-vmware.tar.gz  (link alternativo)

Versão Xen Server    - http://www.inf.ufrgs.br/~ffdaitx/scripts-xen.tar.gz            (link principal)
                     - http://sites.google.com/site/fabiodaitx/scripts-xen.tar.gz     (link alternativo)
```

Por questões de organização e espaço o código fonte não foi impresso neste trabalho.

APÊNDICE B METODOLOGIA DE TESTE E DETALHES DOS RESULTADOS

Neste Apêndice são apresentados os códigos fonte utilizados na implementação dos experimentos para o agente proposto. Os testes dos experimentos realizados neste trabalho foram realizados utilizando scripts bash e tcl executados várias vezes a fim de se obter resultados confiáveis dentro do intervalo de confiança de cada teste.

A metodologia utilizada para os testes do experimento 1 e o código correspondente é apresentado a seguir:

Listing B.1: Metodologia Experimental 1

```
# sair do X (interface gráfica)
/etc/init.d/gdm stop
ctrl+alt+f1

#inicializar agente snmp
sudo bash
cd /home/ubuntu/eclipse/workspace/net-snmp-5.5
./recompila

#executar testes
export MIBS+=VIRTUAL-ROUTER-MIB
/home/ubuntu/Desktop/script(01-29).sh

#verificar conectividade (validar)
pingar de uma ponta (Assinante #1) a outra (Servidor) para ver se está roteando
opcionalmente, lançar iperf para gerar tráfego (unicast ou dual)
iperf.exe -s -P 0 -i 1 -p 5001 -f k      # Servidor: Escuta na porta 5001
iperf -c 192.168.0.102 -r -L 5000 -t 60  # Cliente: Unicast TCP para -> Servidor
iperf -c 192.168.0.102 -d -L 5000 -t 60  # Cliente: Dual TCP <-> Servidor

#re-fazer os testes 10x
logar no servidor https://vmware:8333
parar VM e deletar

#recompila (re-inicializa) para o agente recuperar o estado
./recompila

################################################################################

Scripts utilizados (VMware Server apenas):

script(01-29).sh: implementa os testes e coleta o uso de CPU e Memória
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/script(01-29).tar.gz               (link principal)
  - http://sites.google.com/site/fabiodaitx/script(01-29).tar.gz        (link alternativo)

invoca ...

router(1-2)_script.sh: invoca primitivas e coleta o tempo de resposta de cada uma
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/router(1-2)_script.tar.gz          (link principal)
  - http://sites.google.com/site/fabiodaitx/router(1-2)_script.tar.gz   (link alternativo)

invoca ...

conf_router(1-2).tcl: configura cada roteador virtual
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/conf_router(1-2).tar.gz            (link principal)
  - http://sites.google.com/site/fabiodaitx/conf_router(1-2).tar.gz     (link alternativo)
```

O *script* "recompila"foi apresentado no Apêndice A.4. Os códigos disponibilizados para realização dos testes só devem ser reutilizados respeitando as condições de uso da licença GNU GPL (General Public License).

A metodologia utilizada para os testes do experimento 2 e o código correspondente é apresentado a seguir:

Listing B.2: Metodologia Experimental 2

```
# sair do X (interface gráfica)
# para o VMware:
/etc/init.d/gdm stop
ctrl+alt+f1
# para o Xen:
acessar linha de comando

# executar testes para 10 roteadores virtuais (case10)
sudo bash
cd <pasta do script case10.sh>
./case10.sh

################################################################################

Scripts utilizados (VMware Server e Xen Server):

auth_info.sh: informações especificas por plataforma (usuários, pastas, etc.)
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/auth_info.sh                (link principal)
  - http://sites.google.com/site/fabiodaitx/auth_info.sh         (link alternativo)

utilizado por ...

case10.sh: implementa os testes e coleta o uso de CPU e Memória
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/case10.sh                   (link principal)
  - http://sites.google.com/site/fabiodaitx/case10.sh            (link alternativo)

invoca ...

router_script.sh: invoca primitivas e coleta o tempo de resposta de cada uma
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/router_script.sh            (link principal)
  - http://sites.google.com/site/fabiodaitx/router_script.sh     (link alternativo)

invoca ...

removal_script.sh: invoca primitiva para remover vr e coleta tempo de resposta
Download:
  - http://www.inf.ufrgs.br/~ffdaitx/removal_script.sh           (link principal)
  - http://sites.google.com/site/fabiodaitx/removal_script.sh    (link alternativo)
```

APÊNDICE C ARTIGO PUBLICADO – IM 2011

Neste anexo é apresentado o artigo entitulado *"On the use of SNMP as a Management Interface for Virtual Networks"*. Este trabalho foi publicado nas sessões técnicas da conferência IM 2011, como um artigo completo. O *paper* apresenta a interface de gerenciamento para redes virtuais. A solução proposta é avaliada utilizando-se um agente SNMP executado sobre as plataforma Xen e VMware. A avaliação experimental considerou diversos parâmetros, como tempo de execução, consumo de CPU, consumo de memória e impacto no tráfego.

- Título: *On the use of SNMP as a Management Interface for Virtual Networks* ("Sobre a utilização de SNMP como uma Interface de Gerenciamento para Redes Virtuais")
- Conferência: *IFIP/IEEE International Symposium on Integrated Network Management* (IM 2011)
- URL: http://www.ieee-im.org/
- Data: 23-27 de Maio de 2011
- Local: Dublin, Irlanda

On the use of SNMP as a Management Interface for Virtual Networks

Fábio Fabian Daitx, Rafael Pereira Esteves, Lisandro Zambenedetti Granville
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, Brazil
Email: {ffdaitx, rpesteves, granville}@inf.ufrgs.br

***Abstract*—Virtual networks emerged as an alternative for network infrastructure provision. However, the growth in users demand for shared resources over network elements increase allocation complexity. Thus, enabling effective management on each node is key to the global network operation. This work investigate the use of SNMP as a management interface for virtual routers in order to support automation and provide optimal use of physical assets. The solution was evaluated in two virtualization platforms (Xen and VMware) in order the verify the advantages and limitations of the proposed management interface.**

I. Introduction

Network virtualization [1] emerged as an alternative to help the development of new network architectures and to then overcome the so-called Internet ossification [2]. In network virtualization, the physical infrastructure, called substrate, is shared among multiple virtual networks, each of them employing its own protocols, addressing scheme, and policies in an independent and isolated way.

Unlike server/host virtualization, which usually happens only at the edge nodes of a networked environment, network virtualization takes place in the core of the network. The term ”network virtualization”, in fact, may be used to refer to different things. One of them, which is key to this paper, is the notion that virtual routers run on top of physical routers. One single physical router can host several different virtual ones, enabling several overlay networks running simultaneously over a single physical infrastructure. Like in server virtualization, network virtualization creates a more flexible environment where, for example, virtual routers could migrate from one physical router to another. This flexibility leads to the general notion (although not widely accepted) that network virtualization will be a core component of the Future Internet [3].

Although the research on network virtualization is quite active today, few investigations are found regarding the management of physical routers which are able to host virtual ones. In practice, most of the management actions over such devices require manual intervention via command line interfaces. Even though this can be sufficient for the administration of current, small scale network substrates, it is certainly not sufficient for the management of the Future Internet environment.

In this paper we focus on the following research questions on the management of network virtualization: (1) what are the management requirements of physical and virtual routers; (2) what is a possible design for a management interface of physical and virtual routers; and (3) what is the performance of current, off-the-shelf physical and virtual routers implementing the management interface considered in question 2. The contribution of our work comes from the fact that, as far as the authors of this paper are aware of, no other research has addressed the management issues of routers for network virtualization beyond mentioning that management primitives would be required.

As a starting point, we propose a management solution based on the Simple Network Management Protocol (SNMP). Our objective is to investigate the benefits and limitations of a network virtualization management interface based on SNMP and thus to deal with the boundaries of the traditional management for virtual networks. In this paper, we are specifically interested in the interfacing problem in order to list basic operations that a management interface must support in a virtualized environment. We believe that our observations are important in the sense that they can help identifying new requirements that have to be matched by a management interface in a network virtualization context. We propose MIB (*Management Information Base*) extensions to represent virtual network resources and define basic management operations. To achieve this, we used the Net-SNMP agent as a basis for our extensions and Vyatta as a router solution running on top of virtualization platforms, like Xen and VMware Server.

The remaining of this paper is organized as follows. In Section 2 we review the related work on network virtualization listing the most significant researches carried out so far. The model of a generic physical router that supports network virtualization is presented in Section 3, where its associated management requirements are discussed. In Section 4 we present relevant options for a management interface, in addition of showing an implementation based on off-the-shelf solutions for virtual routers. The performance of the proposed interface is then evaluated and the results achieved are presented in Section 5. Finally, we close this paper in Section 6, where final remarks and directions for future work are discussed.

II. Background

The term “network virtualization” is often used to address different subjects loosely related to each other [1]. “Network virtualization”, itself, usually refers to a network environment where Internet Service Providers (ISPs) split their roles into Infrastructure Providers (InP) and Service Providers (SP). Sometimes, this is also called “Infrastructure as a Service”

(IaaS) [4], where technologies like VPNs and Overlay Networks are investigated. The concept of "link virtualization" refers to multiplexing techniques used to share links between multiple packet flows. For instance, different kinds of virtual circuit techniques (such as those employed on Frame Relay, ATM, and MPLS) could be addressed in this context. One can refers to "virtualized networks" when talking about virtual network testbeds like Planetlab [5], OneLab [6], and GENI [7]. Finally, "node virtualization" is presented as a software technique used to make a single physical device behave as (or look like) multiple instances of a real device [8].

Virtual routers (or logical routers) are isolated software implementations of physical routers. They have been used since the early 2000s to implement Layer 3 Provider Provisioned Virtual Private Networks (L3 PPVPN) for site-to-site communications. In this approach, the Provider Edge (PE) device contains a virtual router (with logically independent routing domains) per VPN and no extra disambiguate address mechanism is needed for VPN reachability [9]. A backbone implementation using this methodology has been deployed in SINET3 network [10].

The emulation of complete network environments has been enabled by new technologies such as VMware [11] and Xen [12]. Virtual machines and virtual routers can be used to implement such environments at both the network edges and core, respectively. This work is particularly interested in the management of emulated virtual routers, which work close to commercial off-the-shelf virtual routers currently available on retail.

A technical study about low level implementation aspects of virtual routers can be found in the work carried out by Comer and Martynov [13]. It embraces the development of actual routers and enables a deeper comprehension about their configuration. Some primitives for the management of such devices have been developed recently, such as live router migration [14] [15]. The objective is to make the procedure of migrating or replacing a physical router faster and to minimize disruption by using a remote API method call requested by the network operator. Cisco IOS XR interface actually provides only 2 commands to manage its devices virtualization features [16]: one to create and the other to assign a virtual router to a physical device. Juniper Junos IOS has an interface [17] that enables full configuration of virtual routers following a procedure very similar to the one used on traditional routers.

However, none of these systems provide a complete interface covering all the needed operations that must be present in a network virtualization environment. These operations can include virtual router creation, virtual network interface creation and configuration, virtual router initialization, virtual router removal, virtual network interface removal and so on.

An approach of Management Information Base (MIB) for Virtual Routers was partially implemented as an IETF draft [18]. This MIB was developed in the context of L3 VPNs, but was discontinued. It is composed of objects related to the high-level configuration of virtual routers, as well as objects that collect statistics about these devices. The main problem with this MIB is that it is not suitable for operations like router migration and flexible network interface bindings.

III. Proposal of a Management Interface for Virtual Routers

Since SNMP is widely used for network management, it becomes a natural alternative for handling routers when considering interoperability, simplicity, and reuse. The flexibility of decoupling hardware and software brought by virtualization technologies came along with the increased complexity that can be created when several large scale networks are overlaid. In this context, end users interact with the services offered by service providers to build and run network applications suitable for their needs. The main operations that end users can perform are registering in a specific virtual network and requesting for more resources (*e.g.*, bandwidth). Service providers, in turn, act as resource providers to other entities (*i.e.*, other service providers) and, thus, the creation of hierarchical virtual resources is possible, which allows the construction of more large and complex virtual network infrastructures. The need for a high degree of automation in the management of such environments can be primarily enabled using SNMP agents running in physical routers offering an appropriate management information base (MIB) that can be used by advanced management applications.

A. Reference Model

This work is focused on defining how virtual routers (VRs) can be managed through SNMP. Most operations for virtual hosts are the same as the ones defined for virtual servers such as creation, removal, and configuration of virtual resources. These resources encompass the whole managed device or parts of its hardware (*e.g.*, network interface cards).

One single physical router with n ($n \geq 1$) network interfaces can emulate one or more virtual routers with m ($m \geq 1$) network interfaces each. An example of architecture for virtual routers have been introduced by the VROOM proposal [14]. In this paper we present an extension of that model in order to incorporate a management interface for network devices (Figure 1).

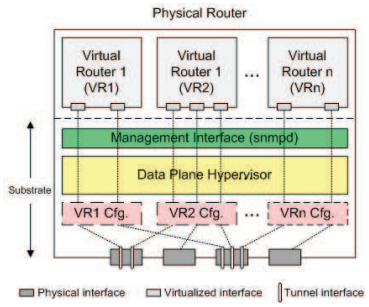


Fig. 1. Manageable virtual router architecture.

A physical router is conceptually divided in two layers: a low-level layer, or substrate, where a global view of the device is available and data plane features are executed, and a high-level layer, where isolated virtual routers implement their control plane. The management support is also split in two planes: in the low level, global functions (as virtual router creation, removal, and migration, as well as network interface

mappings) are performed by an SNMP agent (snmpd), and a thin layer of software (hypervisor) runs on top of the application being used to control the hardware; while in the high level, control plane activity (*e.g.*, routing protocols) is handled. In this approach, each virtual router contains 2 configuration levels: a low level configuration (VR1 Cfg., VR2 Cfg., etc.) that controls network interface bindings, and a high level configuration (internal to each virtual router) that controls routing and other control plane related features.

In essence, in this scenario, there are two entities that need to be managed: the substrate and the different virtual routers hosted above it. Two approaches can be employed in the the management of a physical router that support virtualization:

i) **A core, single agent to manage the whole system -** In this approach, a single SNMP agent, placed in the substrate (as shown in Figure 1), can manage both the hosting physical router and the hosted virtual routers. Network management communication is established between the remote manager and this global agent, which can then access both low layer and high layer MIBs. Addressing is done using SNMP contexts [18] [19] defined in SNMP community strings or SNMPv3 context names. For example, SNMP messages sent to a device that hosts two VRs (having their own contexts "vr01" and "vr02") are used to access "vr01", "vr02", and the substrate (*e.g.*, "admin").

ii) **Multiple agents -** in this case, each managed entity (both physical and virtual routers) contains a full SNMP agent that offers its own particular MIB. Network management communication is then established between the manager and each virtual and physical router. Addressing is accomplished through common UDP/IP mechanisms, since physical and virtual devices have their own network address. Messages are directly received and processed by each agent.

In any case, when *snmpd* receives a request message from the manager, it issues a command to the system hypervisor in order to retrieve or set SNMP variables. An API (Application Programming Interface) must then be available at the hypervisor side in order to enable the SNMP agent operations. As an alternative, in case of unavailable API, the physical device could also be managed by editing some of its configuration files, but that would significantly increase the agent implementation complexity. This section is devoted to the core management issues and focus on the SNMP agent and its associated MIB for virtualization management through the interaction of the physical router's hypervisor.

B. Virtual Router Extended MIB

Basically, virtual routers management require a set of information that can be classified into device configuration, network interface bindings, and device statistics. In this work, the Virtual Router MIB [18] originally defined in an Internet draft is used as a basis for our proposed management interface. Although an expired draft is taken into account, we prefer this approach of starting with a previously defined (yet obsolete) MIB than writing a new one from scratch.

The original Virtual Router MIB covers some fundamental operations such as creation and removal of virtual routers, as well as the binding between a virtual and physical network interfaces. The original set of MIB objects, however, is not sufficient to cope with all operations required for the basic management of a virtual network. For example, the Virtual Router MIB does not support binding more than one virtual interface to a single physical interface at the same time. In order to improve the original Virtual Router MIB, we propose a Virtual Router Extended MIB organized according to Figure 2. The highlighted objects were added to the original Virtual MIB in order to allow flexible network interface binding.

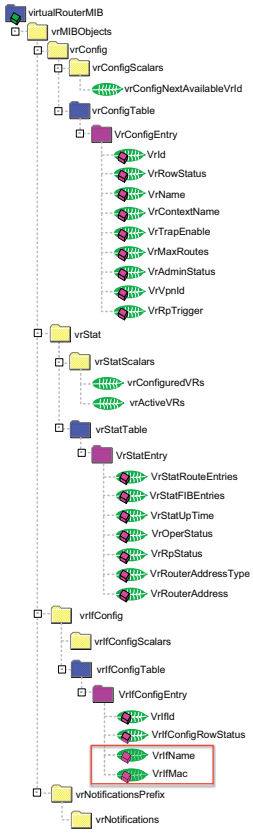


Fig. 2. Virtual Router Extended MIB.

The **`vrConfig`** group contains information about VR indexing, creation, and deletion of VRs. To aid the management station in assigning a new VR without conflict, the management station can retrieve the next available VR identifier accessing the `vrConfigNextAvailableVrId` scalar from the managed device. Each virtual router configuration is stored in the `vrConfigTable`, which contains information as the virtual router identifier (`vrId`), name (`vrName`) and status (`vrRowStatus`). The status column is used to create and delete virtual routers. Other information in this table refer

to internal configuration of each virtual device and include the Virtual Private Network Identifier (vrVpnId), the maximum number of virtual routes supported (vrMaxRoutes), and a trigger to initiate or shutting down routing protocols (vrRpTrigger).

The **vrStat** group contains scalar objects with global device information (*e.g.*, vrConfiguredVRs - the number of VRs configured in each node - and vrActiveVRs - the number of VRs that are active on the network element) and a table with the administrative and the operating status of each VR (vrStatTable). This table holds statistics as the current number of route entries (vrStatRouteEntries), the IP address of one of the VR interfaces (vrRouterAddress), and the elapsed time since the VR became operational (vrStatUpTime).

The **vrIfConfig** group includes a table for configuring VR interfaces (vrIfConfigTable). This table holds all the bindings between the virtual routers and the physical network interfaces. As in the vrConfigTable, virtual router interfaces can be created, deleted, or having their operational status modified by editing the vrIfConfigRowStatus variable. This column is used to trigger the creation and removal operations or to set the operational state for each of the VR virtual interfaces. In order to set a new VR interface, the SNMP agent first checks whether the corresponding vrId exists and then checks if the chosen vrIfId is available. The vrIfName and vrIfMac objects are used to define additional parameters of the virtual interface and to allow the binding between a physical and a virtual interface.

The **vrNotifications** group allows the manager to enabled or disabled traps by handling the vrTrapEnable variable from the vrConfigTable. These traps include vrUp, vrDown, and vrMaxRoutesExceeded.

C. Example operations

In this subsection we exemplify how the variables of our proposed extended MIB are manipulated in order to execute some operations for the management of virtual routers.

1) Virtual router creation: For virtual router creation, the variables vrRowStatus and vrName are used in the following way.

- Retrieving the next available Virtual Router ID using the vrConfigNextAvailableVrId to create a VR:

```
GetRequest {vrConfigNextAvailableVrId.0}
Response   {vrConfigNextAvailableVrId.0 = 9}
```

- In vrConfigTable, create a VR instance using vrRowStatus:

```
SetRequest {
   vrRowStatus.9  createAndGo(4),
   vrName.9  "TestVR"
}
```

Additional parameters can also be set in when creating a new VR, such as the context name (vrContextName), the administrative status (vrAdminStatus), and a flag to enable traps (vrTrapEnable). On success, the new virtual router is created and started, and the vrConfigNextAvailableVrId variable is incremented by 1.

2) Virtual router removal: For virtual router removal, we use the vrRowStatus variable:

```
SetRequest {vrRowStatus.9  destroy(6)}
```

This action involves stopping the virtual router before removing it. This can lead to an extended delay in performing a VR removal.

3) Dynamic interface binding: Managing virtual network interfaces is another fundamental aspect that has to be covered in order to build a management solution for router virtualization. As for virtual routers, virtual network interfaces will be created, configured and destroyed. Specifically, virtual network interfaces have to be associated to one physical network interface. This association is called interface binding or mapping. In order to support virtual interfaces creation, two variables have been added to the vrConfigTable:

- The **vrIfName** variable stores the name of the physical network interface card (NIC) (*e.g.* “eth0”, “eth1”), being responsible for the actual binding between the physical and the logical interface. The name is set during interface creation and can be re-configured later.
- The **vrIfMac** variable stores the MAC address for the virtual interface. User can specify it manually or use the generate special value; in this case, the agent will automatically allocate a new unused MAC address. If not specified during interface creation, vrIfName will assume the generate value by default.

An example of creation of a VR interface instance using vrIfConfigRowStatus from the vrIfConfigTable is:

```
SetRequest {
    vrIfConfigRowStatus.1.3  createAndGo(4),
               vrIfName.1.3  "NAT_vmnet8",
                vrIfMac.1.3  "generate"
}
```

In this case, the agent will trigger the creation of a new interface identified with the value 3 for the virtual router 1 and activate it (up); this operation should success even when the router is running (*hot plugging*). If the specified virtual router does not exist or other interface identified with value 3 already exists, the operation will fail. Both vrIfConfigRowStatus and vrIfName are mandatory parameters during the VR interface creation, so if any of them is not present in the varbind list, the operation will fail. Dynamic binding can be achieved by setting the virtual router interface name, for example:

```
SetRequest {vrIfName.1.3  "Bridged_eth0"}
```

Finally, removal is performed by setting the instance's row status:

SetRequest {vrIfConfigRowStatus.1.3 destroy(6)}

Although SNMP is not commonly used for configuration purposes, ignoring it as a management interface for network virtualization would not be appropriate either. Our proposal is then a starting point to identify general requirements for management interfaces in network virtualization. Recently, NETCONF protocol [20] emerged as a standardized alternative for network configuration. Using NETCONF as management interface for virtual router configuration is not well investigated up to now.

IV. Case study and evaluation

In this section, we present a case study of the proposed management interface to verify its feasibility in a real network scenario. Here, we are considering feasibility as the easiness of the use of the proposed interface to manage virtual network resources. To achieve that, VMware Server [11] and Xen [12] have been used as the virtualization platforms on which the management interface has been deployed and evaluated. In order to evaluate the impact over an SNMP agent's performance, these platforms, acting as the substrate of a physical router, have been used to host an increased number of virtual routers.

Management of virtual routers and management of virtual machines are not the same, even though they share similarities. The difference lays in the fact that virtual routers have specific objects related to their routing services. Some of these primitives are very similar to virtual machines, such as creation and removal of virtual routers, but others are not, such as virtual network interfaces creation/removal and router/link migration.

To individually implement each virtual router, it was used a solution where *software routers* turn inexpensive PCs into network routers. Such software routers are composed of daemons that, for example, support routing protocols like OSPF, RIP, and BGP. In this way, they can fully implement a router's control plane.

In this investigation, we have used Vyatta [21] because of its easiness to use and configure, and given the fact that it is available for free.

A. SNMP agent implementation

A new MIB module has been coded and attached to an SNMP agent that handles the requests and executes the operations defined in section III-B. The proposed implementation was based on the Net-SNMP 5.5 package [22]. The agent has been integrated through a command line interface (*bash scripting*) with the underlying API functions offered by the virtualization platform hypervisor. Xen uses the *"xe"* command to manage its virtual resources (*e.g.*, virtual machines, disks, and interfaces), while VMware uses both *"vmrun"* and *"vmware-vim-cmd"*. Actions to create virtual elements are triggered by SNMP *set request* messages. SNMP messages, at the agent side, result in executing *bash* scripts that contact the hypervisor and the operating system or edit some configuration file.

Network interface binding has been accomplished using some API functions and directly editing the virtual machine configuration files. Directly editing configuration files is not a proper solution since it increases complexity and may lead to unexpected behaviors, but it has been used because of hypervisor's API limitations. The `vrIfName` object (Section III-C3) is the name that corresponds to the combination between the network type (*bridged, host only, nat*) and the physical interface (*eth0, eth1, ..., ethN*). When requested to edit or to create a new virtual network interface card, the SNMP agent checks if the received parameters from the varbind list are sufficient and valid.

B. Case study: network consolidation

To evaluate the proposed solution, the implemented SNMP agent has been used in the following scenario. Network consolidation [23] consists on using physical routers that combine features from multiple real routers in a single device. Consolidation can be deployed at network edge and core and can group equipments from different levels of the routing hierarchy (*vertical*) or from the same level (*horizontal*). Horizontal edge consolidation can be used to provide access for wide area network (WAN) services for different subscribers through Internet service provider's (ISP) points of presence (POPs).

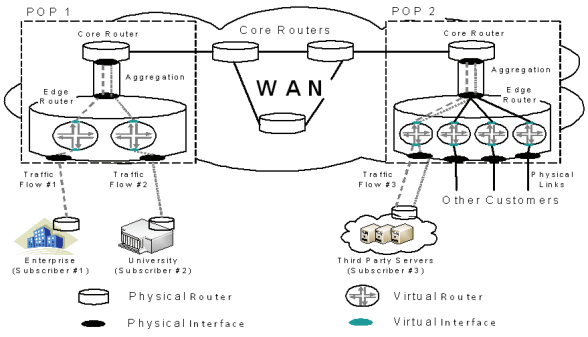


Fig. 3. Case study of horizontal edge consolidation management.

We assumed the situation depicted in Figure 3, where two ISP clients (subscriber #1 and subscriber #2) need to access a remote site service (say, they are streaming some media or clouding its servers), generating the traffic flows #1 and #2, respectively. POPs tend to add routers in order to supply new services or to provide connectivity for new subscribers when their network edge devices are running out of capacity. New architectures can deal with these scenarios using virtualization. In this case, several edge physical routers can be replaced by a single device hosting several virtual routers. By reducing physical devices and interfaces, power, space and installation requirements real operational and capital savings can be achieved. Horizontal edge consolidation management can be achieved with manual intervention via command line interface (CLI). However, as the number of subscribers and network size increase, it becomes harder to collect and analyze the status of multiple devices in order to properly engineer the network. In this way, the use of SNMP can be an

adequate solution for managing virtual routers since it allows a faster and uniform way of triggering and capturing device configuration and statistics.

In addition to network consolidation, the literature presents many other cases where network management primitives are used to deal with network virtualization. Planned maintenance [14] [15] is mainly focused on link and virtual router migration. SINET3 [10] presented a network architecture that uses virtualization in the core (vertical core consolidation) to implement multiple service networks (on demand service networks can be created and edited with VR creation and VR removal). Large scale network testbeds [5] [6] [7] use node virtualization (similar to horizontal core consolidation) to provide large scale connectivity (experiments may need changing virtual network interface bindings, for example).

C. Experiment description

The experiment presented in this section intend to evaluate our proposed solution by testing the implemented SNMP agent operations and by analysing different performance parameters. The presented case study (Section IV-B) is used as reference for the testbed creation. Tests have been carried out over a single network edge device running the SNMP agent and receiving network and management traffic. Tests have been conducted in three steps:

1) Creating the first VR: in this case, the substrate has no load and its capacity is fully available;
2) Routing traffic with the first VR: after the creation, the first VR is put into production sending/receving network traffic from subscriber #1 and routing it to/from a remote server (Figure 4);

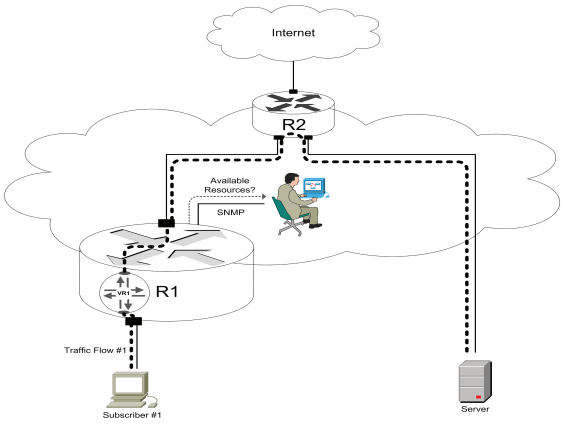


Fig. 4. Manager monitoring system status.

3) Creating a new VR: in this case, the substrate has both VR1 deployed and its routing operation load (Figure 5);

At a first moment, the network has no user accessing network through R1. When a new demand arrives, the manager requests this router to deploy the first virtual router (VR1) so that connectivity can be provided for subscriber #1. Once connected, subscriber #1 can start routing traffic through VR1.

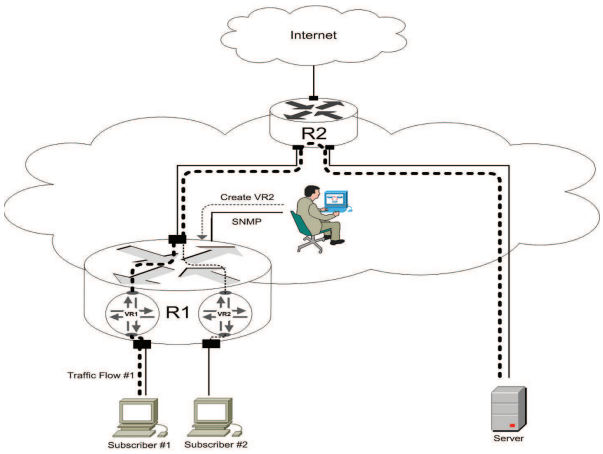


Fig. 5. Manager set a new virtual router creation.

The subscriber network administrator should migrate links or activate devices to direct its traffic (traffic flow #1) to this new virtual router.

Later, when a new virtual router is required once again, the manager should consider new parameters that are now available such as NICs load, CPU, and memory usage (Figure 4). These R1 statistics will determine resource availability for deploying a new virtual router (VR2). Finally, the manager can safely run the command to create a new VR (Figure 5).

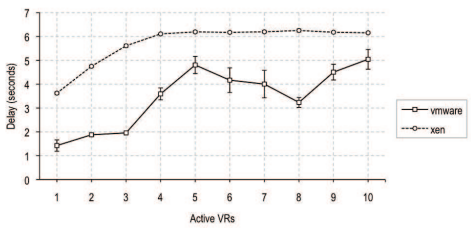
The sequence of management operations to create any new VR is:

1) Create VR;
2) Create first virtual network interface;
3) Create second virtual network interface;
4) Start VR;
5) Wait VR boot time;
6) Configure VR.

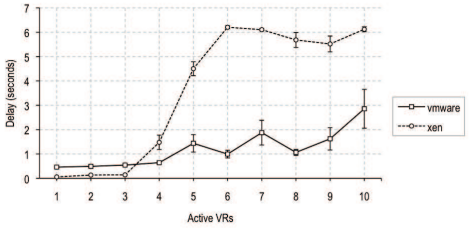
The experiments have been carried over an Intel® Core^{TM} i7-920, 2.66 GHz processor with 6 GB of RAM acting as R1, and two conventional PCs to transmit/receive traffic ("subscriber #1" and "server"). VRs have been implemented with the Vyatta 6.0 software router. Finally, for R1 substrate, VMware Server 2.0.2-203138 and Xen Server 5.5 have been tested.

D. Results

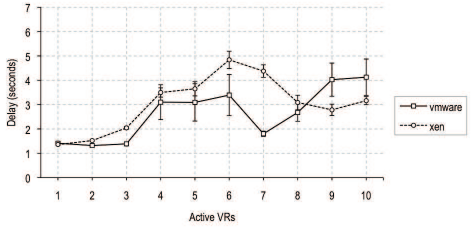
The average delay, perceived at the manager side, for the operations of VR creation, virtual network interface creation, VR start, and VR removal have been computed. The time to provide connectivity for subscribers is critical for ISPs business, since they can only start charging customers when the service is available. In our observations, we have measured the impact of creating successive VRs. Each new VR n has been created over the physical router that was already hosting $n-1$ VRs. VRs varied from 1 to 10. Figure 6 presents the results considering 30 experiments with a confidence interval of 95%.



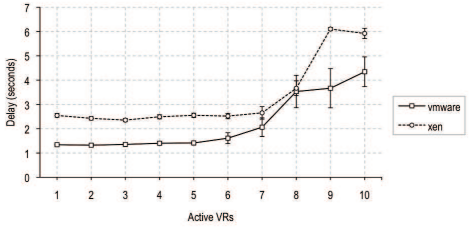
(a) Response time for VR creation



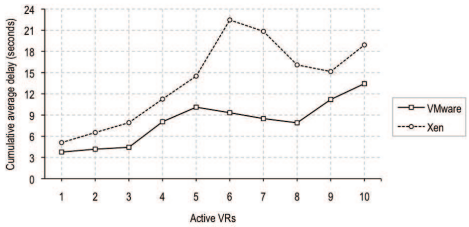
(b) Response time for the creation of a virtual network interface



(c) Response time for VR start

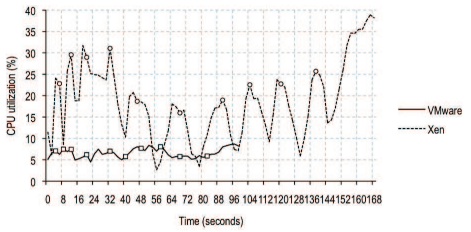


(d) Response time for VR removal

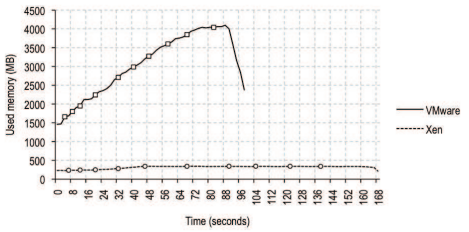


(e) Accumulated response time for VR operations

Fig. 6. Response time for VR operations



(a) CPU load



(b) Used memory

Fig. 7. Total CPU load and used memory for VR operations

It is possible to observe that, in a general way, VR operations in VMware Server perform better than in Xen Server. For VR creation (Figure 6(a)), the response time tends to stabilize after the creation of the fourth VR in Xen. In VMware there are variations in the times obtained for a different number of active VRs because, in this case, resource consumption (*e.g.*, CPU, and memory) changes more frequently through time.

For the creation of a virtual network interface (Figure 6(b)) Xen is better than VMware for 3 or less active VRs. For 4 or more VRs, VMware presents a much lower response time. Virtual network interface creation time in Xen tends to lay between 5 and 6 seconds for 5 or more active VRs.

VR start is an instruction that triggers the initialization of a VR. For this operation (Figure 6(c)), VMware is faster than Xen until 7 active VRs. After that, Xen turns out to perform faster than VMware. In the case of VR removal (Figure 6(d)), VMware still has a superior performance than Xen. In Figure 6(e), the total time for the VR operations is presented. Here, it can be noticed that VMware is faster than Xen for executing the whole set of VR operations.

We also observed the CPU and memory usage for VR operations (Figure 7(a) and Figure 7(b)), with 2 seconds sampling interval. CPU and memory observation is critical for the allocation of new virtual routers.

The market dots in the curves represent the instants when a VR is created and put operational (VR creation, two virtual network interfaces creation, and VR start) in both platforms. There is a difference on the number of collected samples of VMware and Xen. This is due to the fact that, as stated before, in Xen the response time for the VR operations is higher than in VMware, thus the total sampling time in Xen is also higher.

For the CPU load, it can be noticed that VMware has a lower CPU consumption than Xen. This is not necessarily due only to the execution of the VR operations, since there are other processes that influence these values. On the other hand, memory utilization in Xen is lower than VMware.

From these results, it is possible to conclude that the fact that CPU consumption is higher in Xen influences the response time of the VR operations in this platform. In this way, we can state that response time of VR operations is more related to the CPU load level of the machine at a given moment than to the memory usage.

V. Conclusion and Future Work

In this paper we presented a study on the use of SNMP for the management of virtualized network resources, more specifically, virtual routers. It has been demonstrated that SNMP can be used to manage scenarios of network virtualization, avoiding the need of manually interacting with the network equipments via CLI and, thus, providing an uniform management interface that can be adapted for use in different (hardware) platforms.

Regarding the virtualization platform evaluated, VMware Server demonstrated a superior performance compared to Xen in terms of response time and CPU utilization, while Xen performed better for memory utilization. It has been observed that response time is more influenced by the CPU load than by the memory usage.

SNMP is not traditionally used for configuration. However, an SNMP-based management interface turns out to be a good start point in order to identify requirements for a management interface for virtual networks. In a general way, a management interface has to support basic operations like creation and configurations of virtual network resources (*e.g.*, virtual routers, virtual network interfaces) and it has to access the low level functions of a specific platform supporting virtualization.

A drawback of the use of SNMP in virtual networks is that the variable that need to be defined for each SNMP agent implementation are highly coupled with the specific characteristics of the underlying infrastructure (platform). Therefore, even if the same set of basic messages can be used to manage the virtual network resources, some agents can require additional objects (e.g., to support router migration) and, thus, present different performance levels. Another aspect related to this is that a full configuration of a virtual router usually requires the handling of several objects which can impact the feasibility and the performance if the number of managed virtual routers is high.

As future work, we intend to develop, evaluate, and compare other management interfaces based on more recent management technologies like NETCONF and Web Services. We also plan to extend this proposed management interface to include configuration of virtual hosts and virtual links.

References

[1] N. M. K. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse through Virtualization," *Computer*, vol. 38, no. 4, Apr. 2005.

[3] P. Papadimitriou, O. Maennel, A. Greenhalgh, A. Feldmann, and L. Mathy, "Implementing Network Virtualization for a Future Internet," in *20th ITC Specialist Seminar on Network Virtualization*, May 2009.

[4] M. Lemay, "An Introduction to IaaS Framework," Aug. 2008, http://www.iaasframework.com/.

[5] (2010, Apr.) Planetlab - an open platform for developing, deploying, and accessing planetary-scale services. [Online]. Available: http://www.planet-lab.org/

[6] (2010, Apr.) Onelab - future internet test beds. [Online]. Available: http://www.onelab.eu/

[7] (2010, Apr.) Geni - global environment for network innovations. [Online]. Available: www.geni.net

[8] S. Maier, D. Herrscher, and K. Rothermel, "Experiences with node virtualization for scalable network emulation," *Comput. Commun.*, vol. 30, no. 5, pp. 943–956, Mar. 2007.

[9] H. O.-B. Paul Knight *et al.*, "Network based IP VPN Architecture using Virtual Routers," draft-ietf-ppvpn-vpn-vr-04, May 2003.

[10] S. Urushidani, K. Fukuda, Y. Ji, S. Abe, M. Koibuchi, M. Nakamura, S. Yamada, K. Shimizu, R. Hayashi, I. Inoue, and K. Shiomoto, "Resource Allocation and Provision for Bandwidth/Networks on Demand in SINET3," in *Network Operations and Management Symposium (NOMS) Workshops*, Apr. 2008, pp. 212–218.

[11] (2010, Apr.) Vmware. [Online]. Available: http://www.vmware.com

[12] (2010, Apr.) Xen hypervisor. [Online]. Available: http://www.xen.org/

[13] D. Comer and M. Martynov, "Building experimental virtual routers with network processors," in *Testbeds and Research Infrastructures for the Development of Networks and Communities - TRIDENTCOM 2006*, Mar. 2006, pp. 9 pp.–230.

[14] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 231–242, Oct. 2008.

[15] M. Agrawal, S. R. Bailey, A. Greenberg, J. Pastor, P. Sebos, S. Seshan, K. van der Merwe, and J. Yates, "RouterFarm: Towards a Dynamic, Manageable Network Edge," in *SIGCOMM Workshop on Internet Network Management*, 2006, pp. 5–10.

[16] (2009, Aug.) Cisco logical routers. [Online]. Available: http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html

[17] (2009, Aug.) Juniper logical routers. [Online]. Available: http://www.juniper.net/techpubs/software/junos/junos85/feature-guide-85/id-11139212.html

[18] E. Stelzer, S. Hancock, B. Schliesser, and J. Laria, "Virtual router management information base using smiv2," draft-ietf-ppvpn-vr-mib-05, 2003.

[19] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing snmp management frameworks," RFC 3411, 2002.

[20] J. Schönwälder, M. Björklund, and P. Shafer, "Network configuration management using NETCONF and YANG," *Comm. Mag.*, vol. 48, pp. 166–173, September 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1867013

[21] (2009, Aug.) Vyatta open networking. [Online]. Available: http://www.vyatta.com

[22] (2009, Aug.) Net-snmp. [Online]. Available: http://www.net-snmp.org/

[23] "Intelligent logical router service," White Paper, Juniper, 2004.