

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

PAULO ROBERTO DA PAZ FERRAZ SANTOS

**Gerenciamento de Roteadores Virtuais em
Ambientes de Virtualização de Redes
Heterogêneos**

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

Orientador: Prof. Dr. Lisandro Granville
Zambenedetti

Porto Alegre
2015

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Paulo Roberto da Paz Ferraz

Gerenciamento de Roteadores Virtuais em Ambientes de Virtualização de Redes Heterogêneos / Paulo Roberto da Paz Ferraz Santos. – Porto Alegre: PPGC da UFRGS, 2015.

120 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Orientador: Lisandro Granville Zambenedetti.

1. Gerenciamento de redes. 2. Virtualização de redes. 3. Roteadores virtuais. 4. Interface de gerenciamento. I. Zambenedetti, Lisandro Granville. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento

Diretor do Instituto de Informática: Prof. Luis da Cunha Lamb

Coordenador do PPGC: Prof. Luigi Carro

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“A persistência é o menor caminho do êxito.”

— CHARLES CHAPLIN

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus, por me dar forças e saúde, essenciais durante toda a trajetória dessa jornada da minha vida.

Meus sinceros agradecimentos aos meus pais, que me deram a base, o incentivo e o bom exemplo para que eu pudesse superar as dificuldades e alcançar meus objetivos.

Um especial e carinhoso agradecimento à minha tão amada esposa Sheila Mayumi, pelo suporte incondicional, pela compreensão e infinita paciência, sem os quais seria impossível eu chegar até o fim dessa desafiadora empreitada.

Meus emotivos agradecimentos à minha filha Daniela e ao meu filho Rafael, que nasceram durante essa longa jornada, 2012 e 2014, e me abençoaram com muita alegria e amor, mesmo tornando essa jornada ainda mais difícil e desafiadora. Papai os ama muito!

Não posso deixar de agradecer aos chefes, atual e anterior, do 1º Centro de Telemática de Área, Organização Militar onde trabalho, pela autorização e liberação para a realização do sonhado mestrado. Agradeço também ao Major João, chefe da Divisão de Operação, por me apoiar e entender as dificuldades de conciliar trabalho e estudo.

Agradeço ao Prof. Lisandro Z. Granville, por me aceitar como orientando, pela valiosa orientação, pela paciência e compreensão para com a minha limitação de tempo, e por ter acreditado e investido em mim. Não menos importante, agradeço ao Rafael P. Esteves pela grande ajuda, principalmente pelas revisões, críticas e sugestões, com paciência e dedicação, e por todas as vezes que cedeu seu tempo para dar dicas e tirar dúvidas quando solicitado.

Agradeço também aos meus outros professores da UFRGS, Paulo M. Engel, Liane M. R. Tarouco e Luciano P. Gaspar, pelos importantes ensinamentos transmitidos em suas aulas. Digo-lhes que estes conhecimentos foram de muita valia no mestrado e, continuam sendo, na minha carreira profissional. Agradeço novamente ao Prof. Luciano, pois foi ele quem inicialmente acreditou no meu potencial e me indicou para o Prof. Lisandro.

Por fim, aproveito a oportunidade para agradecer a todos os amigos que fiz na UFRGS durante todo esse período, dentre eles destaco: Ricardo dos Santos, Oscar Caicedo, Matheus Cadori, Maicon Kist, Pedro Isolani, Juan Wagner e Renata Neuland. Em especial, meu profundo agradecimento ao Ricardo, ao Oscar e ao Cadori, pelas incontáveis vezes que me ajudaram com dicas preciosas e oportunas, sem as quais meu caminho teria sido muito mais longo e sinuoso. Muito obrigado!

RESUMO

Em ambientes de virtualização de redes (NVEs – *Network Virtualization Environments*), a infraestrutura física é compartilhada entre diferentes usuários (ou provedores de serviços) que criam múltiplas redes virtuais (VNs – *Virtual Networks*). Como parte do provisionamento de VNs, roteadores virtuais (VRs – *Virtual Routers*) são criados dentro de roteadores físicos que suportam a virtualização. Atualmente, o gerenciamento de NVEs é quase sempre realizado por soluções proprietárias, normalmente baseadas em interfaces de linha de comando (CLI – *Command Line Interface*). NVEs heterogêneos (*i.e.*, com equipamentos e tecnologias diferentes) são difíceis de gerenciar, devido à falta de soluções de gerenciamento padronizadas. Como primeiro passo para conseguir a interoperabilidade de gerenciamento, bom desempenho e alta escalabilidade, foram implementadas, avaliadas e comparadas cinco interfaces de gerenciamento de roteadores físicos que hospedam roteadores virtuais. As interfaces são baseadas em SNMP (v2c e v3), NETCONF, e RESTful Web Services (sobre HTTP e HTTPS), e são projetadas para realizar três operações básicas de gerenciamento de VRs: criação de VR, recuperação de informações de VR e remoção de VR. Essas interfaces foram avaliadas em relação às seguintes métricas: tempo de resposta, tempo de CPU, consumo de memória e uso da rede. Os resultados mostram que a interface baseada no SNMPv2c é a mais adequada para pequenos NVEs, sem rigorosos requisitos de segurança, e o NETCONF é a melhor escolha para compor uma interface de gerenciamento para ser implantada em cenários mais realistas, onde a segurança e a escalabilidade são as principais preocupações.

Palavras-chave: Gerenciamento de redes. virtualização de redes. roteadores virtuais. interface de gerenciamento.

Virtual Router Management in Heterogeneous Network Virtualization Environments

ABSTRACT

In network virtualization environments (NVEs), the physical infrastructure is shared among different users (or service providers) who create multiple virtual networks (VNs). As part of VN provisioning, virtual routers (VRs) are created inside physical routers supporting virtualization. Currently, the management of NVEs is mostly realized by proprietary solutions, usually based on Command Line Interfaces (CLI). Heterogeneous NVEs (*i.e.*, with different equipment and technologies) are difficult to manage due to the lack of standardized management solutions. As a first step to achieve management interoperability, good performance, and high scalability, we implemented, evaluated, and compared five management interfaces for physical routers that host virtual ones. The interfaces are based on SNMP (v2c and v3), NETCONF, and RESTful Web Services (over HTTP and HTTPS), and are designed to perform three basic VR management operations: VR creation, VR information retrieval, and VR removal. We evaluate these interfaces with regard to the following metrics: response time, CPU time, memory consumption, and network usage. Results show that the SNMPv2c interface is the most suitable one for small NVEs without strict security requirements and NETCONF is the best choice to compose a management interface to be deployed in more realistic scenarios, where security and scalability are major concerns.

Keywords: network management, network virtualization, virtual router, management interface.

LISTA DE ABREVIATURAS E SIGLAS

#VR	Quantidade de Roteadores Virtuais
AES	Advanced Encryption Standard
AMS	Autonomic Management System
API	Application Programming Interface
AUTOI	Autonomic Internet
BEEP	Blocks Extensible Exchange Protocol
CFB	Cipher Feedback
CH	Clearinghouse
CLI	Command Line Interface
CM	Component Manager
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DoS	Deny of Service
FEDERICA	Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures
FIB	Forwarding Information Base
GENI	Global Environment for Network Innovations
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IETF	Internet Engineering Task Force
INM	In-Network Management
InP	Infrastructure Provider
IP	Internet Protocol

ISP	Internet Service Provider
JAXB	Java Architecture for XML Binding
JSON	JavaScript Object Notation
JSSE	Java Secure Socket Extensions
L3VPN	Layer 3 Virtual Private Network
LSP	Label Switched Path
MPLS	Multiprotocol Label Switching
MIB	Management Information Base
NAT	Network Address Translation
NETCONF	Network Configuration Protocol
NOC	Network Operation Center
NVE	Network Virtualization Environment
NVP	Network Virtualization Platform
OID	Object Identifier
Op. Grc.	Operação de Gerenciamento
OpenYUMA	Open YANG Unified Modular Automation
PC	Personal Computer
QoS	Quality of Service
RA	Resource Agent
RCLI	Remote Command Line Interface
REST	Representational State Transfer
RPC	Remote Procedure Call
RSpec	Resource Specification
RWS	RESTful Web Services
SA	Slice Authority
SDH	Synchronous Digital Hierarchy

SDN	Software-defined Networking
SEs	Self-managing Entities
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SNMPv2c	Simple Network Management Protocol version 2c
SNMPv3	Simple Network Management Protocol version 3
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SONET	Synchronous Optical Networking
SP	Service Provider
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UCLP	User Controlled Lightpaths
UDP	User Datagram Protocol
UML	Unified Modeling Language
UPB	User Policy Board
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USM	User-based Security Model
UUID	Universally Unique Identifier
VAS	Virtual Access Server
vCNS	vCloud Network and Security
vCPI	virtualization Component Programmability Interface
VLAN	Virtual Local Area Network

VM	Virtual Machine
VMI	Virtualization Management Interface
VMM	Virtual Machine Monitor
VMM-MIB	Virtual Machine Manager - Management Information Base
VN	Virtual Network
VNARMS	Virtual Network-based Autonomic network Resource control and Management System
VNO	Virtual Network Operator
Vnode	Virtual node
VNP	Virtual Network Provider
VNRM	Virtual Network Resource Manager
VPN	Virtual Private Network
VR	Virtual Router
VR-MIB	Virtual Router - Management Information Base
vSPI	virtualization System Programmability Interface
XML	eXtensible Markup Language
YANG	Yet Another Next Generation

LISTA DE FIGURAS

Figura 2.1 Ambiente de Virtualização de Redes (NVE).....	19
Figura 2.2 Princípios de arquitetura em um NVE.....	20
Figura 2.3 Modelo de gerenciamento conceitual para virtualização de redes	22
Figura 3.1 Modelo de gerenciamento do 4WARD.....	25
Figura 3.2 Arquitetura do AUTOI.....	27
Figura 3.3 Aprovisionamento de fatias no FEDERICA.....	29
Figura 3.4 Principais entidades do ProtoGENI.....	31
Figura 3.5 Estrutura do UCLP	32
Figura 3.6 Aprovisionamento de VNs no VNARMS	33
Figura 3.7 Arquitetura SDN.....	34
Figura 3.8 Arquitetura do Switch OpenFlow	35
Figura 3.9 Esquema de virtualização de redes do VMWare NSX	37
Figura 3.10 Interação entre componentes básicos do VMWare NSX.....	38
Figura 3.11 Grupos de informações da VR-MIB.....	45
Figura 3.12 Alocação de recursos virtuais	47
Figura 3.13 Estrutura da VMM-MIB	48
Figura 3.14 Tabelas da VMM-MIB	49
Figura 3.15 Objetos de notificação da VMM-MIB.....	50
Figura 4.1 Arquitetura de um roteador físico que hospeda VRs.....	52
Figura 4.2 Estrutura da NEW-VMM-MIB com novos objetos e tabelas	54
Figura 4.3 Código da tabela <code>vmConfigTable</code> : SMIV2 <i>versus</i> YANG.....	56
Figura 4.4 Diagrama de classes do modelo de dados para o armazenamento da tabela <code>vmConfigTable</code> em um Map	57
Figura 4.5 Fluxo de mensagens da interface de gerenciamento baseada no SNMPv2c	58
Figura 4.6 Fluxo de mensagens da interface de gerenciamento baseada no SNMPv3	60
Figura 4.7 Fluxo de mensagens da interface de gerenciamento baseada no NETCONF	61
Figura 4.8 Fluxo de mensagens da interface de gerenciamento baseada no RWS	62
Figura 4.9 Fluxo de mensagens da interface de gerenciamento baseada no RWS sobre HTTPS	63
Figura 5.1 Consumo de memória do XenServer com VRs ativos.....	70
Figura 5.2 Tempo de resposta das interfaces de gerenciamento	71
Figura 5.3 Tempo de CPU das interfaces de gerenciamento	73
Figura 5.4 Consumo de memória das interfaces de gerenciamento.....	75
Figura 5.5 Uso da rede pelas interfaces de gerenciamento	76
Figura 5.6 Situação de inversão entre SNMPv2c e NETCONF	77

LISTA DE TABELAS

Tabela 3.1 Classificação quanto aos alvos de gerenciamento	41
Tabela 3.2 Classificação quanto às funções de gerenciamento	42
Tabela 3.3 Classificação quanto à abordagem de gerenciamento	42
Tabela 4.1 Classificação dos projetos e das interfaces propostas quanto aos alvos de gerenciamento.....	63
Tabela 4.2 Classificação dos projetos e das interfaces propostas quanto às funções de gerenciamento	64
Tabela 4.3 Classificação dos projetos e das interfaces propostas quanto à abordagem de gerenciamento	65
Tabela 5.1 Tempo de resposta médio das interface de gerenciamento.....	79
Tabela 5.2 Tempo de CPU máximo das interfaces de gerenciamento	80
Tabela 5.3 Consumo médio de memória das interfaces de gerenciamento.....	80
Tabela 5.4 Tráfego médio gerado pelas interfaces de gerenciamento.....	81
Tabela 5.5 Ordem crescente de uso da rede das interfaces de gerenciamento para operações de criação de VRs	81
Tabela 5.6 Ordem crescente de uso da rede das interfaces de gerenciamento para operações de remoção de VRs	82
Tabela 5.7 Ordem crescente de uso da rede das interfaces de gerenciamento para operações de recuperação de informações de VRs.....	82
Tabela 5.8 Classificação das interfaces de gerenciamento de acordo com as métricas avaliadas	83

SUMÁRIO

1 INTRODUÇÃO	14
2 FUNDAMENTAÇÃO TEÓRICA	18
2.1 Virtualização de Redes	18
2.2 Gerenciamento de NVEs	21
3 TRABALHOS RELACIONADOS	24
3.1 Soluções de Gerenciamento	24
3.1.1 4WARD	24
3.1.2 AUTOI	26
3.1.3 FEDERICA	28
3.1.4 ProtoGENI	30
3.1.5 UCLP	31
3.1.6 VNARMS	32
3.1.7 OpenFlow	34
3.1.8 VMWare NSX	36
3.2 Classificação das Soluções de Gerenciamento	39
3.2.1 Alvos de gerenciamento	39
3.2.2 Funções de gerenciamento	39
3.2.3 Abordagens de gerenciamento	40
3.2.4 Classificação	41
3.3 Gerenciamento de NVEs em ambientes heterogêneos	43
3.3.1 VR-MIB	44
3.3.2 VMM-MIB	46
4 INTERFACES DE GERENCIAMENTO PARA VIRTUALIZAÇÃO DE ROTEADORES	52
4.1 Arquitetura	52
4.2 Modelo de Dados	53
4.3 Fluxo de Mensagens	57
4.3.1 SNMPv2c	57
4.3.2 SNMPv3	59
4.3.3 NETCONF	59
4.3.4 RESTful Web Services	61
4.3.5 RESTful Web Services sobre HTTPS	62
4.4 Classificação	62
5 AVALIAÇÃO	66
5.1 Ambiente de Teste	66
5.2 Metodologia e Métricas	67
5.3 Resultados	69
5.3.1 Tempo de resposta	70
5.3.2 Tempo de CPU	72
5.3.3 Consumo de Memória	74
5.3.4 Uso da Rede	74
5.4 Comparação	78
6 CONCLUSÃO	85
6.1 Principais Contribuições e Resultados Obtidos	86
6.2 Considerações Finais e Possíveis Investigações Futuras	87
REFERÊNCIAS	89
ANEXO A ARTIGO PUBLICADO – WGRS 2014	96
ANEXO B ARTIGO PUBLICADO – IM 2015	111

1 INTRODUÇÃO

A Internet tem sido incrivelmente bem sucedida na modelagem da nossa forma de acesso e troca de informações no mundo moderno. No entanto, sua popularidade se tornou o maior impedimento para o seu maior crescimento. Devido à sua natureza multifornecedora, a adoção de uma nova arquitetura ou a modificação da existente exige consenso entre as partes concorrentes (CHOWDHURY; BOUTABA, 2010). Como resultado, as alterações na arquitetura da Internet tornaram-se restritas a simples atualizações incrementais em vez de mudanças radicais. No núcleo da Internet a rigidez para mudanças é ainda maior e, embora isso mantenha o núcleo simples, dificulta o desenvolvimento de novas aplicações. Por esses motivos diz-se que a Internet está ossificada (HANDLEY, 2006).

A diversificação das aplicações e o intenso uso da Internet como infraestrutura global de comunicação acarreta a necessidade de uma série de adições de protocolos e mecanismos à sua arquitetura, a fim de que sejam providas funcionalidades inexistentes na pilha TCP/IP original (ALKMIM; BATISTA; FONSECA, 2011). Nesse contexto, a virtualização de redes surgiu como alternativa viável para auxiliar no desenvolvimento de novas arquiteturas de rede e para ajudar a superar a ossificação da Internet (ANDERSON et al., 2005). Ao contrário da virtualização de servidores, que normalmente está localizada nos nós de borda de um ambiente de rede, a virtualização de redes ocorre principalmente no núcleo (DAITX; ESTEVES; GRANVILLE, 2011).

O ambiente de virtualização de redes (NVE – *Network Virtualization Environment*) tem como característica a dissociação das funcionalidades existentes no modelo de negócio clássico. O papel dos Provedores de Internet tradicionais (ISP – *Internet Service Provider*) é dividido em duas entidades independentes: Provedores de infraestrutura (InP – *Infrastructure Provider*), que gerenciam a infraestrutura física, e Prestadores de Serviços (SP – *Service Provider*), que criam redes virtuais, agregando recursos de vários InPs e oferecendo serviços fim-a-fim (CHOWDHURY; BOUTABA, 2009). Em um NVE a infraestrutura física de vários InPs, chamadas de substrato, é compartilhada entre diferentes usuários (ou prestadores de serviços) que criam múltiplas redes virtuais (VN – *Virtual Network*), cada uma delas utilizando protocolos, esquemas de endereçamento e políticas independentes. Dessa forma, roteadores virtuais (VR – *Virtual Router*) são criados e hospedados dentro de roteadores físicos, possibilitando que várias redes independentes funcionem simultaneamente, de forma isolada, sobre uma única infraestrutura física (CHOWDHURY; BOUTABA, 2009). Ao desvincular os prestadores de serviços dos provedores de infraestrutura e permitir que múltiplas arquiteturas de redes heterogêneas

coabitem em um substrato físico compartilhado, a virtualização de redes proporciona flexibilidade, promove a diversidade, e possibilita maior segurança e capacidade de gerenciamento (CHOWDHURY; BOUTABA, 2009).

Nos últimos anos, o gerenciamento de NVEs começou a receber atenção especial da comunidade de pesquisa, motivando uma série de projetos e artigos. Gerenciar os NVEs é fundamental para garantir o bom funcionamento da infraestrutura física, das redes virtuais hospedadas, e dos serviços suportados pelas redes virtuais (ESTEVES; GRANVILLE; BOUTABA, 2013). No entanto, muitas questões técnicas se interpõem no caminho de sua realização bem sucedida. Um dos principais problemas de gerenciamento a ser resolvido é a definição de uma interface de gerenciamento e protocolo para operar roteadores físicos que hospedam roteadores virtuais.

Interfaces de gerenciamento apropriadas devem permitir que os InPs e SPs possam acessar, operar, manter e administrar os nós e enlaces físicos e virtuais (ESTEVES; GRANVILLE; BOUTABA, 2013). Atualmente, NVEs são operados por meio de interfaces de linha de comando (CLI – *Command Line Interface*) não padronizadas ou através de ferramentas de gerenciamento proprietárias. O gerenciamento de NVEs construídos sobre substratos físicos heterogêneos (*i.e.*, com equipamentos e tecnologias diferentes) fica prejudicado, pois várias ferramentas independentes são necessárias para realizar uma tarefa de gerenciamento, *e.g.*, criar uma rede virtual. Para minimizar a dificuldade de gerenciar um conjunto diversificado de recursos físicos, interfaces de gerenciamento padronizadas devem ser definidas e avaliadas para permitir a interoperabilidade das diferentes plataformas de virtualização com as ferramentas de gerenciamento. Além disso, redes virtuais podem ser construídas a partir de recursos localizados em domínios administrativos distintos (CHOWDHURY; BOUTABA, 2010) e precisam ser gerenciadas adequadamente. Assim, neste contexto, os operadores de NVEs devem optar por uma interface/protocolo que seja capaz de gerenciar de forma eficiente e escalável roteadores físicos que hospedam roteadores virtuais.

Diversos esforços de padronização foram conduzidos pelo IETF (*Internet Engineering Task Force*) para permitir o gerenciamento de VRs em diferentes contextos. Por exemplo, a VR-MIB (STELZER et al., 2005) foi proposta para gerenciar VRs no contexto das *Virtual Private Networks* de camada 3 (L3VPNs). A VRRP-MIB (TATA, 2012) define um VR como uma representação abstrata de um roteador físico e é usada para lidar com falhas. Motivado pelos esforços do IETF, Daitx, Esteves e Granville (2011) propuseram uma solução de gerenciamento de NVE baseada no *Simple Network Management Protocol* versão 2c (SNMPv2c) estendendo a VR-MIB. Apesar do SNMP não ser tradicionalmente usado em tarefas relacionada à configuração, eles demonstraram que o SNMP também pode ser usado no gerenciamento de NVEs.

Nesta dissertação são avaliados outros protocolos como alternativa ao SNMPv2c para o gerenciamento de roteadores físicos que suportam virtualização, ampliando assim a análise apresentada por Daitx, Esteves e Granville (2011). São eles: o SNMP versão 3 (SNMPv3) (CASE et al., 2002), o *Network Configuration Protocol* (NETCONF) (ENNS et al., 2011), o *RESTful Web Services* (RWS¹) e o RWS sobre HTTPS (*Hyper Text Transfer Protocol Secure*). O SNMPv3 foi investigado por causa dos seus recursos de segurança e configuração remota (STALLINGS, 1998), que não estão presentes originalmente no SNMPv2c. O NETCONF e o RWS foram investigados por serem protocolos considerados como possibilidades alternativas ao SNMP para o gerenciamento de redes e por superarem as limitações do SNMP em termos de escalabilidade (HEDSTROM; WATWE; SAKTHIDHARAN, 2011; PRAS et al., 2004). Em vez da tradicional solução de *Web Services* baseada em SOAP (*Simple Object Access Protocol*), o RWS foi escolhido por ser uma solução mais leve e por ser amplamente utilizado em implementações que utilizam a tecnologia *Web Services* na atualidade. No entanto, como o NETCONF e o RWS não são normalmente usados no contexto da virtualização de roteadores, as questões relacionadas ao gerenciamento no domínio dos NVEs exigem uma análise mais aprofundada. O RWS sobre HTTPS foi investigado, nesse contexto, devido aos seus recursos de segurança.

Além disso, nesta dissertação é proposto um modelo de dados atualizado como alternativa à VR-MIB, tendo em vista que esta MIB (*Management Information Base*) não progrediu no caminho da padronização e permanece na situação de proposta expirada desde 2006, deixando a área sem uma solução baseada no SNMP. Inspirada na recente VMM-MIB (ASAI et al., 2014), este trabalho propõe uma nova MIB, chamada NEW-VMM-MIB, alinhada com os atuais esforços do IETF para o gerenciamento de ambientes virtualizados. Para cada protocolo investigado nesta dissertação, foi desenvolvida uma interface de gerenciamento correspondente, e estas foram comparadas utilizando-se três operações básicas de gerenciamento: criação de VR, recuperação de informações de VR e remoção de VR. As interfaces de gerenciamento foram avaliadas em relação às seguintes métricas: tempo de resposta, tempo de CPU, consumo de memória, e uso da rede. Estas interfaces foram implementadas usando-se o Net-SNMP (NET-SNMP, 2015) para o SNMP, o OpenYUMA (GITHUB, 2015) para o NETCONF, e o Jersey (JERSEY, 2015) para o RWS. Utilizou-se o *hypervisor*² XenServer (XENSERVEN, 2015) para emular um roteador físico com suporte à virtualização e a suíte de roteamento Vyatta (BROCADE, 2015) para implementar os VRs.

¹Nesse trabalho será usado RWS como acrônimo de RESTful Web Services.

²*Hypervisor*, ou Monitor de Máquina Virtual (VMM – *Virtual Machine Monitor*), é uma plataforma que possibilita a criação/execução de máquinas virtuais e controla o compartilhamento de recursos da máquina física (*host*) com as virtuais (*guest*).

O restante desta dissertação está organizada como segue. No Capítulo 2, são apresentados conceitos teóricos de virtualização de redes e mostrada a dinâmica de gerenciamento de NVEs. No Capítulo 3, são apresentados e classificados diversos projetos recentes sobre virtualização que possuem em seu escopo soluções de gerenciamento de NVEs. Ainda no Capítulo 3, são revisadas as propostas VR-MIB e VMM-MIB. No Capítulo 4, a arquitetura de um roteador físico gerenciável que hospeda roteadores virtuais é apresentada, o modelo de dados atualizado é descrito, e o fluxo de mensagem de cada interface de gerenciamento proposta é explicado. O desempenho das interfaces implementadas são avaliados, e os resultados obtidos são apresentados e comparados no Capítulo 5. Por fim, no Capítulo 6 são ressaltadas as principais contribuições deste trabalho, juntamente com a conclusão dos resultados da avaliação, e apresentadas as considerações finais e possíveis investigações futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos sobre a virtualização de redes, como: definição, modelo de negócio, princípios de arquitetura e objetivos de projetos. Além disso, a dinâmica de gerenciamento de NVEs é mostrada em um modelo de gerenciamento conceitual para virtualização de redes que descreve entidades, relacionamentos e operações de gerenciamento tipicamente encontrados em NVEs.

2.1 Virtualização de Redes

Um ambiente de rede suporta a virtualização de redes, se ele permite a coexistência de múltiplas redes virtuais (VNs) no mesmo substrato físico (CHOWDHURY; BOUTABA, 2010). O substrato é definido como um conjunto de recursos físicos na infraestrutura de rede, tais como PCs (*Personal Computer*), *switches* e roteadores. Cada VN em um ambiente de virtualização de redes (NVE) é uma coleção de nós virtuais e *links* virtuais. Pode-se dizer que essencialmente, uma VN é um subconjunto dos recursos da rede física subjacente.

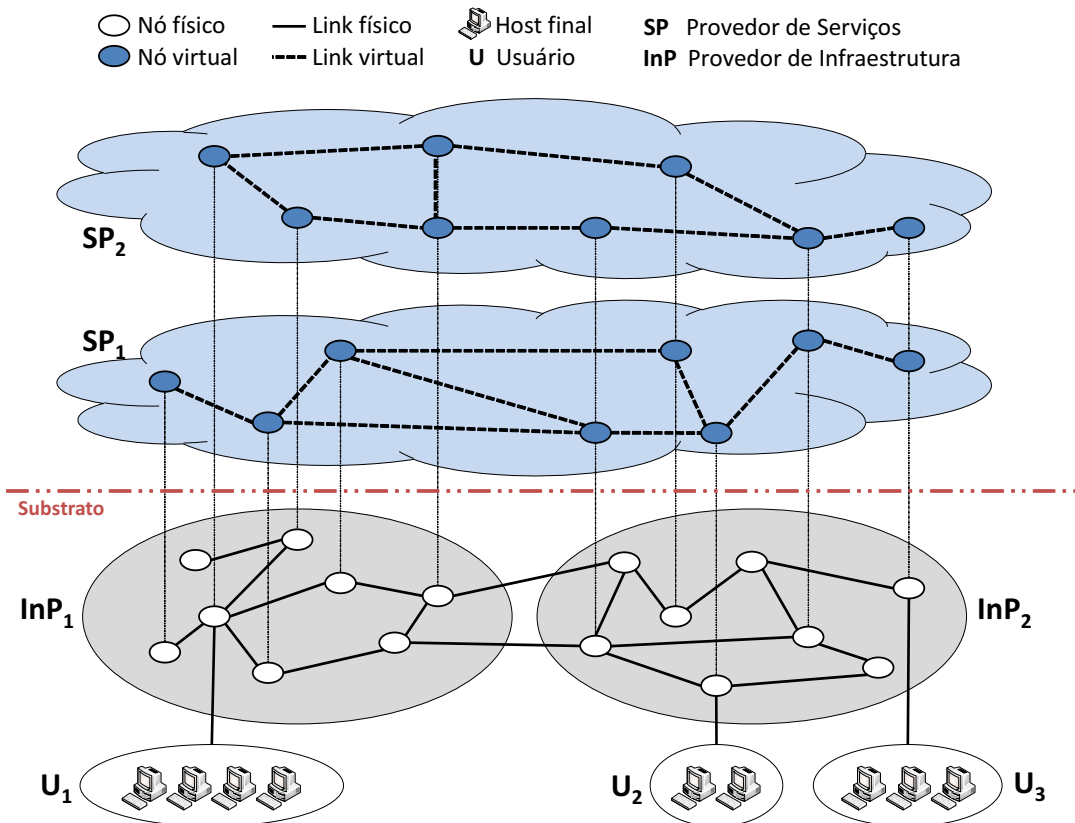
Chowdhury e Boutaba (2009) propuseram que em um ambiente de virtualização de redes (NVE) houvesse a dissociação das funcionalidades do modelo de negócio clássico, separando o papel dos tradicionais Provedores de Serviços de Internet (ISPs) em dois: Provedores de Infraestrutura (InPs) e Prestadores de Serviços (SPs). O primeiro (InP) é responsável por manter a infraestrutura física, enquanto que o último (SP) agrega recursos de vários InPs para criar redes virtuais (VNs) e fornecer serviços de rede para os usuários finais.

Um exemplo de NVE está representado na Figura 2.1. Há, na parte inferior, dois InPs com diferentes topologias físicas e *hosts* finais. Acima deles existem dois SPs, cada um no controle de uma única rede virtual. Como se pode notar, estas redes virtuais têm topologias distintas, tanto entre si quanto em relação à infraestrutura física dos InPs. Na figura, os nós físicos estão representados na cor branca e os nós virtuais na cor azul. Conectando os nós virtuais existem *links* virtuais e, analogamente, conectando os nós físicos, *links* físicos.

Em um NVE podem ser observados 4 princípios de arquitetura, dentro do paradigma da virtualização de redes. São eles: coexistência, recursão, herança e revisitação (CHOWDHURY; BOUTABA, 2009).

- **Coexistência** – A coexistência de várias VNs é a característica definidora de um NVE. Ela se refere ao fato de que várias VNs de diferentes SPs podem coexistir juntas, abrangendo parte ou a totalidade das redes físicas subjacentes fornecidas por um ou mais InPs. Na

Figura 2.1 – Ambiente de Virtualização de Redes (NVE)

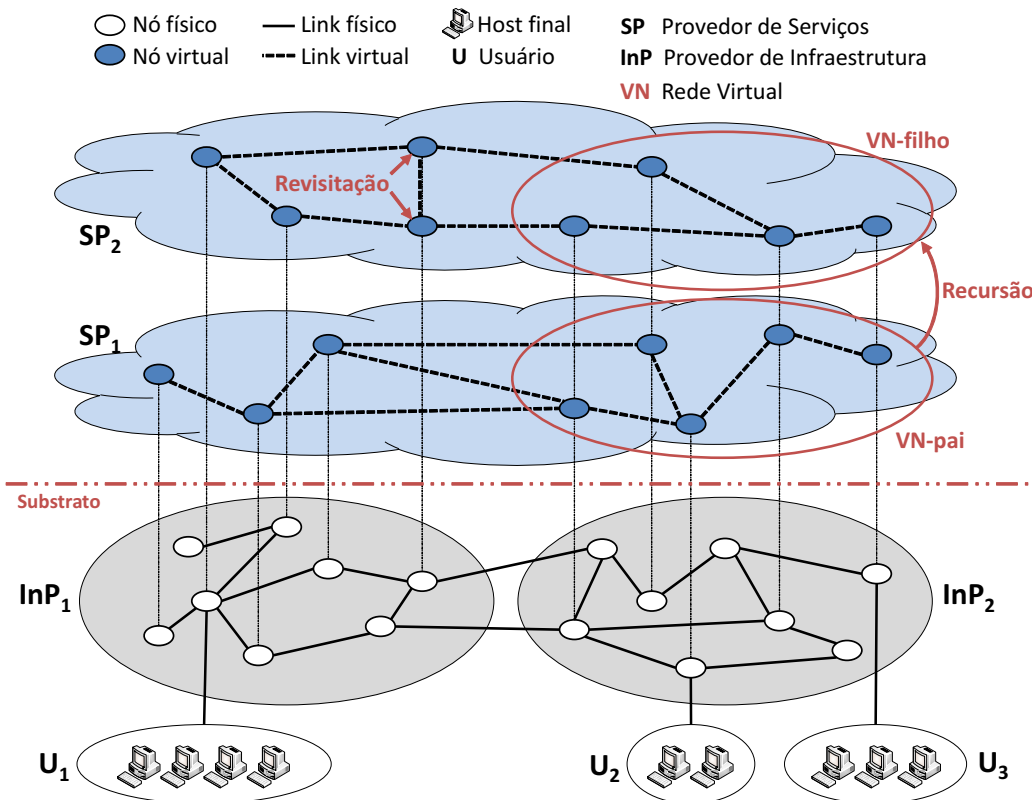


Fonte: CHOWDHURY; BOUTABA (2009).

Figura 2.2, a VN do SP₁ e a VN do SP₂ são duas VNs coexistentes.

- **Recursão** – A recursão, ou aninhamento, existe quando uma ou mais VNs são geradas a partir de outras VN, criando, assim, uma hierarquia de VNs com relações pai-filho. O prestador de serviços SP₁ na Figura 2.2 aluga uma parcela de seus recursos para o SP₂, fazendo o papel de um InP virtual.
- **Herança** – Em um NVE, VNs-filho podem herdar atributos da arquitetura de seus pais, o que também significa que as restrições da VN-pai são traduzidas automaticamente em restrições semelhantes sobre seus filhos. Por exemplo, as restrições impostas pelo InP₂ na Figura 2.2 serão automaticamente transferidas da VN-pai (em SP₁) para a VN-filho (em SP₂) através da herança.
- **Revisitação** – A revisitação permite a um nó físico hospedar vários nós virtuais de uma única VN. O uso de vários roteadores lógicos para lidar com diversas funcionalidades em uma grande rede complexa permite a um SP reorganizar sua estrutura de rede lógica e simplificar o gerenciamento de uma VN. A Figura 2.2 apresenta um exemplo de revisitação na VN do SP₂, conforme indicado.

Figura 2.2 – Princípios de arquitetura em um NVE



Fonte: CHOWDHURY; BOUTABA (2009).

Antigamente os projetos de pesquisa em virtualização de redes concentravam-se em objetivos individuais, tais como segurança, flexibilidade ou programabilidade. Com o passar do tempo, a pesquisa avançou gradualmente e, hoje em dia, os projetos cobrem vários objetivos simultâneos. Chowdhury e Boutaba (2009) definiram alguns critérios que devem ser cumpridos, como objetivos de projeto, a fim de que a virtualização de redes seja materializada com sucesso. São eles: flexibilidade, capacidade de gerenciamento, escalabilidade, isolamento, estabilidade e convergência, programabilidade, heterogeneidade, e suporte ao legado.

- **Flexibilidade** – A virtualização de redes deve fornecer liberdade em todos os aspectos da rede. Cada SP deve ser livre para implementar, de forma independente da rede física subjacente e das demais VNs coexistentes, topologias de rede, funcionalidades de roteamento e encaminhamento de pacotes, protocolos de controle personalizados, etc.
- **Capacidade de gerenciamento** – Ao separar os SPs dos InPs, a virtualização de redes deve modularizar tarefas de gerenciamento de redes e introduzir responsabilidades (*accountability*) em todas as camadas da rede. Deve ser fornecido o completo controle fim-a-fim das VNs aos SPs, eliminando a exigência de coordenação entre os limites administrativos observados na Internet existente.

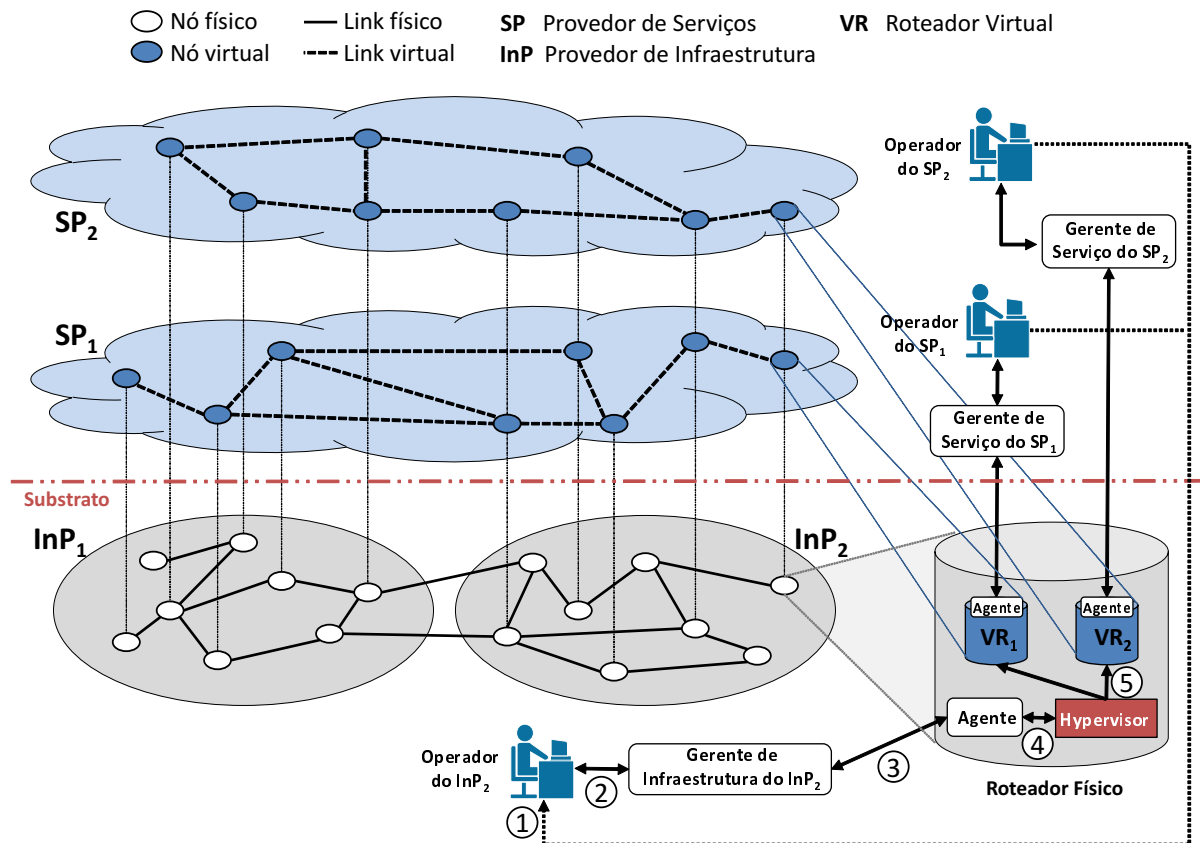
- **Escalabilidade** – InPs em um NVE devem poder escalar para suportar um número crescente de VNs coexistentes sem afetar o seu desempenho.
- **Isolamento** – A virtualização de redes deve garantir o isolamento entre VNs coexistentes para melhorar a tolerância a falhas, segurança e privacidade. Assim, possíveis erros de implementação ou de configuração em uma VN ficam contidos na própria rede e não afetam as outras VNs coexistentes.
- **Estabilidade e convergência** – Apesar de isoladas umas das outras, as VNs estão suscetíveis a falhas na rede física subjacente e podem ficar indisponíveis caso haja instabilidade no InP (*e.g.*, oscilação de roteamento). A virtualização deve assegurar a estabilidade de um NVE, e no caso de qualquer instabilidade, as VNs afetadas devem ser capazes de convergir com sucesso aos seus estados estáveis.
- **Programabilidade** – Através da programabilidade os SPs podem implementar protocolos customizados e implantar serviços diversos. A situação ideal a ser alcançada é uma programabilidade que seja fácil, eficaz e segura ao mesmo tempo.
- **Heterogeneidade** – A heterogeneidade deve ser considerada em todos os níveis: rede física subjacente, VNs e usuários finais. As infraestruturas subjacentes podem ser de diferentes tecnologias (*e.g.*, óptica e *wireless*) e devem ser capazes de suportar protocolos e algoritmos heterogêneos implementados pelos diferentes SPs. Os SPs, por sua vez, devem ser capazes de criar e operar VNs fim-a-fim entre domínios sem depender de uma solução tecnológica específica. Além disso, a heterogeneidade dos dispositivos do usuário final também deve ser considerada.
- **Suporte ao legado** – Suporte ao legado significa manter a compatibilidade com as versões anteriores, e é sempre uma questão de grande preocupação durante a implantação de uma nova tecnologia. Conceitualmente, a virtualização de redes pode facilmente dar o suporte ao legado por considerar a Internet existente como apenas mais uma VN em seu rol de redes, mas se isso pode ser feito de forma eficiente permanece uma questão em aberto.

2.2 Gerenciamento de NVEs

O gerenciamento de NVEs é realizado através de operações de gerenciamento, que podem ser classificadas em operações de gerenciamento de InP e de SP. A primeira (gerenciamento de InP) inclui, por exemplo, o fornecimento de redes virtuais e o monitoramento de recursos físicos, já a segunda (gerenciamento de SP), por sua vez, lida com a operação de redes virtuais e a prestação de serviços a usuários finais.

Para permitir uma melhor compreensão do gerenciamento de NVEs, a Figura 2.3 mostra um modelo de gerenciamento conceitual para virtualização de redes que descreve entidades, relacionamentos e operações de gerenciamento tipicamente encontrados em NVEs.

Figura 2.3 – Modelo de gerenciamento conceitual para virtualização de redes



Fonte: ESTEVES; GRANVILLE; BOUTABA (2013).

Os InPs oferecem seus nós físicos para hospedar os nós virtuais pertencentes aos SPs. Vários nós virtuais podem ser criados e coexistir de forma isolada dentro de um nó físico. Nós físicos são controlados por um gerente de infraestrutura, que, utilizando um protocolo de gerenciamento, troca mensagens com o agente localizado em cada nó físico. Uma vez que novos nós virtuais são criados, eles são gerenciados pelo SP a que eles pertencem. Desse modo, um gerente de serviço se comunica com os agentes associados a cada nó virtual para coletar informações e aplicar ações de gerenciamento. SPs podem alugar recursos de diferentes InPs para construir suas redes virtuais. Os InPs podem estar localizados em diferentes domínios administrativos (ou sistemas autônomos), portanto, é necessário um bom nível de coordenação entre os diferentes gerentes de infraestruturas.

A criação de nós virtuais pode ser feita manualmente pelo operador do SP ou automaticamente pelo operador do InP, utilizando um algoritmo embarcado. Uma vez que o nó físico é

selecionado, o operador do SP solicita a criação de um nó virtual para o operador do InP (passo 1 da Figura 2.3) que, usando o gerente de infraestrutura, instancia o nó solicitado (passo 2 da Figura 2.3). Cada nó físico tem um *hypervisor* que permite a criação de nós virtuais. Quando o agente do nó físico recebe um pedido do gerente de infraestrutura (passo 3 da Figura 2.3), ele se comunica com o *hypervisor* (passo 4 da Figura 2.3) que, em seguida, executa a ação requerida, *i.e.*, criação do nó virtual sobre o nó físico (passo 5 da Figura 2.3). Em geral, *hypervisors* fornecem APIs que permitem que programas externos chamem operações internas, como criação, inicialização e remoção de nó virtual, e executem *scripts* para realizar o ajuste fino da configuração dos recursos virtuais.

Semelhante aos nós virtuais, os enlaces virtuais são criados através de agentes localizados nos nós da rede física. Antes de entrar em contato com o gerente da infraestrutura, o operador do SP especifica as características desejadas do enlace virtual, como nó de origem, nó de destino, e largura de banda. Assim, o gerente da infraestrutura envia a solicitação aos agentes dos nós físicos que hospedam os nós virtuais de origem e de destino para criar o enlace virtual. Enlaces virtuais podem ser estabelecidos por meio de diversas tecnologias, *e.g.*, configuração de VLANs entre os nós físicos que hospedam os nós virtuais ou configuração de túneis utilizando MPLS (*Multiprotocol Label Switching*), LSP (*Label Switched Path*) ou GRE (*Generic Routing Encapsulation*). Para completar a criação de um enlace virtual, interfaces de rede virtuais pertencentes aos nós virtuais de origem e de destino precisam estar vinculadas às suas respectivas interfaces de rede físicas.

O “proprietário” de uma rede virtual é normalmente um operador humano ou uma outra entidade que, na maioria das vezes, não é a proprietária do substrato físico. O gerenciamento de uma rede virtual não deve afetar a gerência do substrato nem a de outras redes virtuais. Por isso, visões isoladas de gerenciamento têm que ser fornecidas para os diferentes operadores em ambas as camadas física (substrato) e de virtualização. O isolamento no plano de gerenciamento é dependente do isolamento no plano de dados e de controle, que é fornecido habilitando-se tecnologias, tais como VLAN, MPLS, e *hypervisors*.

Há muitos desafios técnicos na implementação de NVEs, dentre os quais o gerenciamento tem uma importância especial. O gerenciamento de NVEs é crucial para garantir o correto funcionamento da infraestrutura física, das rede virtuais hospedadas, e dos serviços suportados pelas redes virtuais (ESTEVEZ; GRANVILLE; BOUTABA, 2013). No próximo capítulo serão apresentados alguns projetos que consideram o gerenciamento um requisito fundamental.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados projetos representativos sobre virtualização de redes, que consideram o gerenciamento de NVE um requisito primordial em suas soluções. Os projetos serão classificados de acordo com critérios apresentados de forma a proporcionar uma visão holística e organizada das soluções de gerenciamento aplicáveis a NVEs. Por fim, o gerenciamento de NVEs em ambientes heterogêneos será discutido e, neste contexto, serão apresentadas duas MIBs (*Management Information Base*) que serão utilizadas como inspiração para uma nova proposta de MIB, descrita no capítulo 4, para gerenciamento de roteadores físicos que hospedam roteadores virtuais.

3.1 Soluções de Gerenciamento

Diversos protótipos de virtualização de redes foram projetados visando tecnologias de rede específicas. A motivação por trás desses projetos é explorar características únicas destas redes para permitir a virtualização. Nesta seção, foram selecionados oito projetos em estado avançado de desenvolvimento (ou já concluídos) que representam trabalhos recentes relacionados à virtualização de redes. Os projetos escolhidos tem como característica comum considerarem o gerenciamento de NVE um requisito primordial em suas soluções. São eles: 4WARD, AUTOI, FEDERICA, ProtoGENI, UCLP, VNARMS, OpenFlow e VMWare NSX.

3.1.1 4WARD

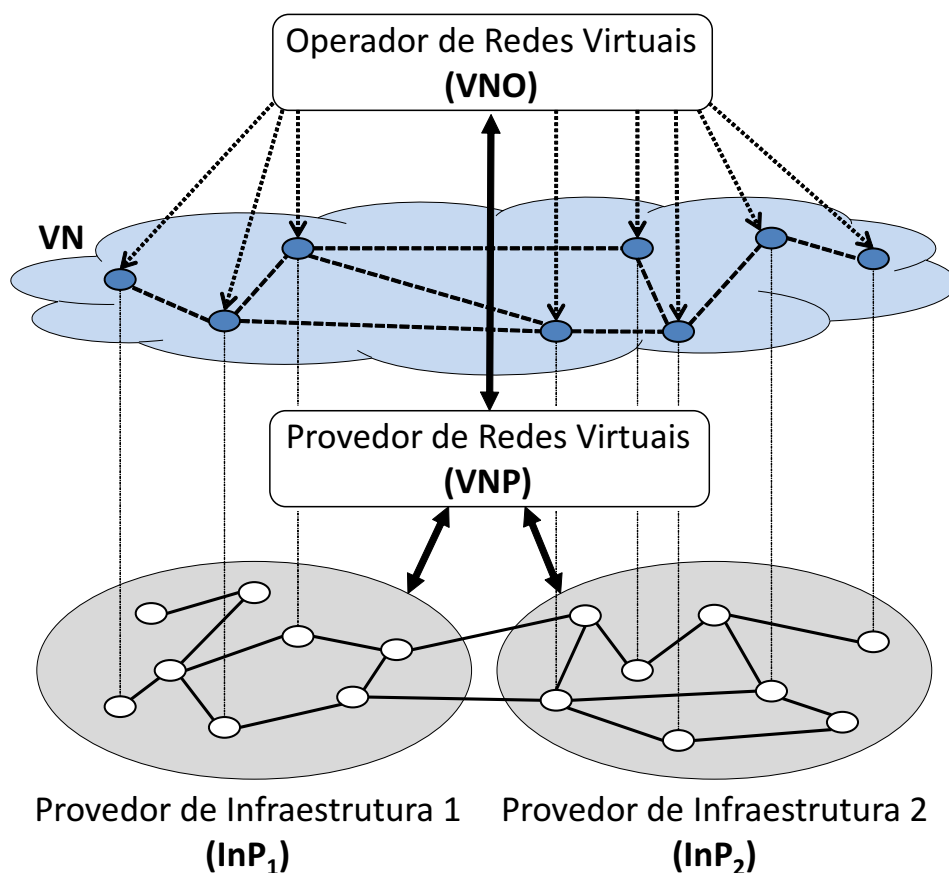
O projeto FP7 4WARD (4WARD, 2015) define uma estrutura de virtualização de redes, chamada VNet, projetada para gerenciar múltiplas redes virtuais coexistentes em uma infraestrutura compartilhada por meio da virtualização de recursos de rede a nível de operadora. Ele fornece meios para apoiar a instanciação sob demanda e a interoperabilidade confiável entre as redes virtuais heterogêneas em um estabelecimento comercial seguro e confiável. 4WARD também suporta a virtualização de tecnologias de rede heterogêneas (*e.g.*, com e sem fio), dispositivos de usuário final heterogêneos, e protocolos de rede novos, como parte do projeto da arquitetura do seu núcleo.

O modelo de negócio 4WARD apresenta três papéis: Provedores de Infraestrutura (InP), que gerenciam os recursos das redes subjacentes; Provedores de Redes Virtuais (VNP – *Virtual Network Providers*), que criam redes virtuais; e os Operadores de Redes Virtuais (VNO – *Virtual Network Operators*), que conectam os clientes com os serviços prestados em diferentes redes

virtuais (CORREIA et al., 2011).

O provisionamento de recursos na VNet inclui a descoberta, a incorporação, e a instanciação. Na fase da descoberta, o provedor da VNet gera uma lista de recursos candidatos a hospedar a rede virtual. O processo de incorporação emprega um algoritmo guloso para definir o mapeamento dos recursos virtuais à rede física. A fase da instanciação consiste em reservar efetivamente os recursos virtuais selecionados. A Figura 3.1 representa o modelo de gerenciamento do 4WARD juntamente com as relações entre as entidades participantes.

Figura 3.1 – Modelo de gerenciamento do 4WARD



Fonte: do autor (2015).

Na VNet, agentes são colocados nos nós físicos para fornecer informações atualizadas sobre os recursos físicos e virtuais para o InP. As informações coletadas são utilizadas para diferentes fins, incluindo a descoberta de recursos e auto-organização das redes virtuais. A VNet também conta com uma estrutura de percepção de situações que agrega informações de monitoramento e oculta detalhes desnecessários para garantir escalabilidade e eficiência do processo de monitoramento.

Além disso, a VNet oferece uma interface de gerenciamento de virtualização baseada em

XML-RPC¹ chamada VMI (*Virtualization Management Interface*), que define um conjunto de operações de gerenciamento, incluindo a criação, a extinção e a concatenação de recursos virtuais. A VNet implementa uma abordagem de gerenciamento distribuído e autônomo, conhecido como INM (*In-Network Management*). No INM, entidades auto-gerenciadas (SEs – *Self-managing Entities*) incorporadas dentro da rede são responsáveis pelo funcionamento autônomo da infraestrutura física.

3.1.2 AUTOI

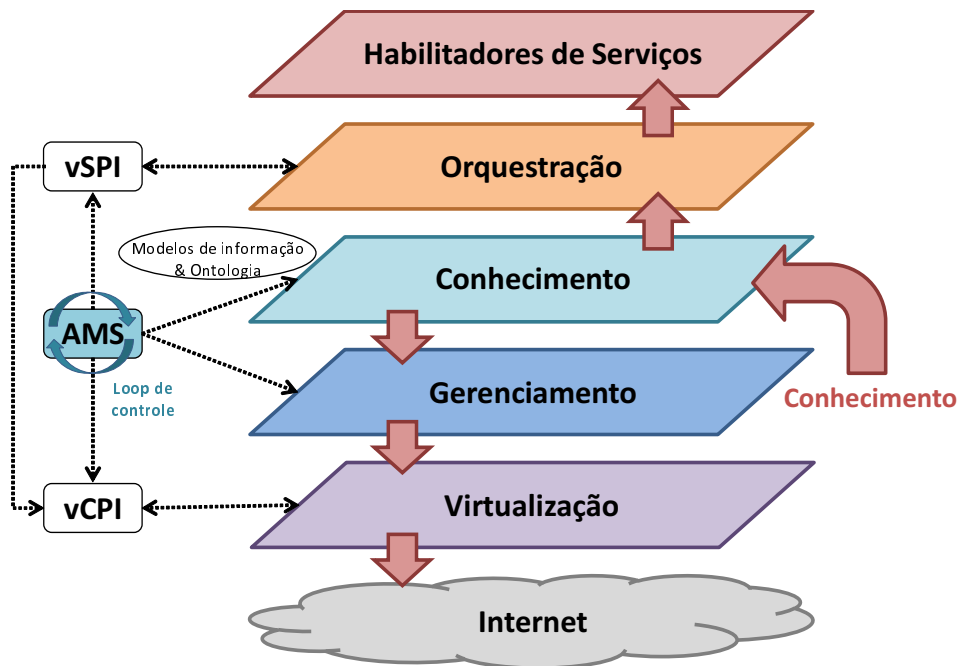
A iniciativa *Autonomic Internet* (AUTOI) (IST-AUTOI, 2015) tem como objetivo desenvolver soluções de gerenciamento autônomo para a Internet do Futuro. A solução AUTOI consiste em sistemas de gerenciamento distribuídos descritos com a ajuda de cinco planos: Habilidade de Serviços, Orquestração, Conhecimento, Gerenciamento e Virtualização (GALIS. et al., 2009). Estes sistemas distribuídos formam uma infraestrutura de controle de rede orientada por software, sendo executados no topo de todas as atuais redes e infraestruturas físicas para fornecer uma sobreposição de recursos virtuais autônomos. Os cinco planos podem reunir observações, restrições e asserções, aplicando-lhes regras para gerar observações e respostas de serviço de modo a convergir para implementações e manutenções otimizadas (RUBIO-LOYOLA et al., 2010). A arquitetura do AUTOI é esquematizada na Figura 3.2.

Cada domínio AUTOI é gerenciado por um Sistema de Gerenciamento Autônomo (AMS – *Autonomic Management System*), que executa um *loop* de controle. Os AMSs são blocos funcionais da solução AUTOI que implementam funcionalidades dos planos de Conhecimento e de Gerenciamento. Através de mapeamento lógico, os AMSs permitem que os dados armazenados em modelos sejam transformados em conhecimento e, em seguida, combinados com o conhecimento armazenado nas ontologias para proporcionar uma estimativa, sensível ao contexto, do funcionamento de um ou mais recursos virtuais. AMSs diferentes podem cooperar uns com os outros, com objetivo de construir serviços fim-a-fim.

O plano de Virtualização do AUTOI é composto de mecanismos de *software* que tratam uma seleção de recursos físicos como um conjunto programável de recursos virtuais. Estes são organizados pelo plano de Orquestração em conjuntos apropriados de recursos virtuais para formar componentes, dispositivos, ou até mesmo redes. AMSs interagem com os planos de Virtualização e de Orquestração através de duas interfaces bem definidas: a Interface de Programa-

¹XML-RPC é um protocolo de chamada de procedimento remoto (RPC – *Remote Procedure Call*) que utiliza XML (*eXtensible Markup Language*) para codificar suas chamadas e o HTTP (*Hyper Text Transfer Protocol*) como um mecanismo de transporte.

Figura 3.2 – Arquitetura do AUTOI



Fonte: do autor (2015).

bilidade de Componente de Virtualização (vCPI – *virtualization Component Programmability Interface*) e a Interface de Programabilidade do Sistema de Virtualização (vSPI – *virtualization System Programmability Interface*).

A vCPI é uma interface que permite o gerenciamento dos recursos virtuais (*e.g.*, roteadores virtuais) de dentro de um componente físico. Ela fornece métodos que suportam a auto-configuração de *links* virtuais e gerenciamento de roteadores virtuais. Métodos de gerenciamento de *links* fornecem suporte para a instanciação, remoção e modificação de *links* virtuais. Métodos de gerenciamento de roteadores virtuais são usados, por exemplo, para registrar, iniciar e desligar um roteador virtual. Ela também implementa métodos que permitem o monitoramento de informações e que são os principais instrumentos para fazer cumprir as funções de gerenciamento de auto-desempenho e falhas.

A vSPI é usada para ativar o plano de Orquestração para governar recursos virtuais, e para construir serviços/redes virtuais que satisfaçam aos objetivos de negócio que possuem requisitos de serviços especificados. Além disso, a vSPI fornece uma visão macro dos recursos virtuais ao plano de Orquestração, que, por sua vez, usa vCPI para construir e gerenciar redes virtuais.

AUTOI utiliza o Lattice como *framework* de monitoramento (CLAYMAN; GALIS; MATAS, 2010). O Lattice baseia-se no conceito de sondas de monitorização e de fontes de dados. Fontes de dados agrupam as informações monitoradas coletadas pelas sondas e enviam para os consumidores interessados, seguindo um modelo de comunicação previamente definido,

tal como publicação/assinatura ou *IP multicast*.

3.1.3 FEDERICA

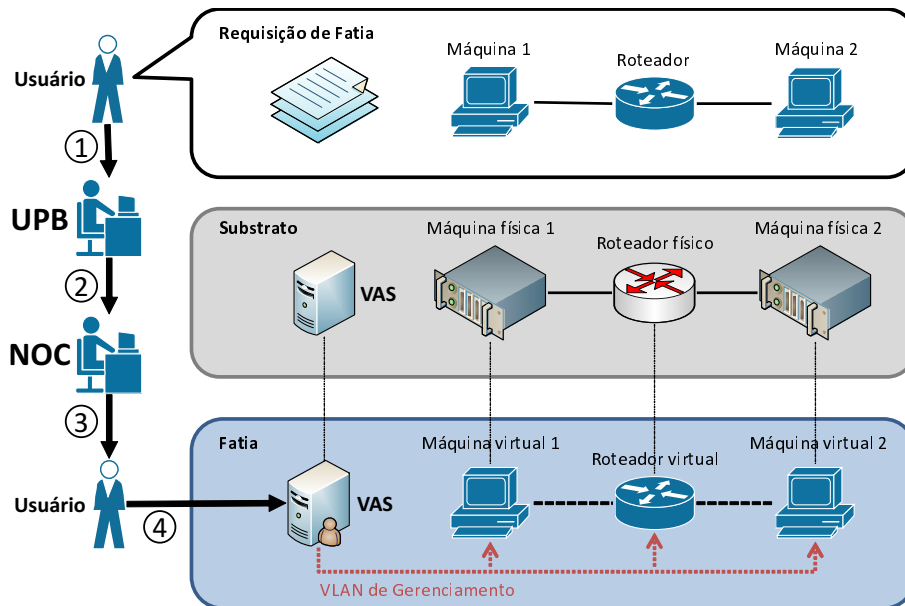
O projeto co-financiado pela Comunidade Europeia FEDERICA (*Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures*) (FEDERICA, 2015) apoia o desenvolvimento da Internet do futuro, por definição, e tem como foco a construção de uma infraestrutura de rede de larga escala para possibilitar a experimentação de novos protocolos de Internet e arquiteturas.

A arquitetura de FEDERICA foi projetada para permitir aos usuários solicitar “fatias virtuais” ou *slices* da infraestrutura, que são na verdade agregações de recursos de redes virtuais (SZEGEDI et al., 2011). Essas fatias são completamente isoladas umas das outras e se comportam exatamente como se fossem uma infraestrutura física, do ponto de vista dos usuários. Além disso, os usuários são capazes de configurar e gerenciar totalmente os recursos de suas próprias fatias, sem que isso afete a operação da infraestrutura física (KRZYWANIA; DOLATA; SZEGEDI, 2010).

FEDERICA possui dois grupos de trabalho: o Conselho de Política de Usuário (UPB – *User Policy Board*) e o Centro de Operações de Rede (NOC – *Network Operation Center*), que são responsáveis pelas operações administrativas e técnicas, respectivamente. O UPB realiza a avaliação inicial de todos os projetos de pesquisa que desejem utilizar ou se conectar à infraestrutura do FEDERICA. Ele garante, essencialmente, que as origens e os destinos sejam acessíveis, as interfaces sejam compatíveis, e a infraestrutura possa lidar com as capacidades necessárias dentro do prazo solicitado. A responsabilidade do NOC é dupla; ele deve garantir a disponibilidade sustentável dos recursos, bem como configurar os recursos físicos para criar fatias virtuais. O processo de provisionamento de uma fatia é apresentado na Figura 3.3.

Quando um usuário necessita de uma fatia, ele envia a requisição de fatia para o UPB incluindo a especificação do experimento planejado (passo 1 da Figura 3.3). A especificação deve incluir restrições de tempo, descrição do trabalho, seus principais objetivos, informações sobre a topologia, e os requisitos de configuração. O UPB analisa a requisição de fatia e, então, aprova ou rejeita. Após a aprovação, a solicitação técnica é passada para o NOC (passo 2 da Figura 3.3). Os administradores do NOC preparam o conjunto de recursos físicos do substrato da infraestrutura, configuram o ambiente da fatia virtual e, em seguida, fornecem a fatia desejada ao usuário solicitante (passo 3 da Figura 3.3). Após o final do tempo de vida da fatia, todos os recursos são liberados e podem, então, ser utilizados para a criação de outras fatias.

Figura 3.3 – Aprovisionamento de fatias no FEDERICA



Fonte: KRZYWANIA; DOLATA; SZEGEDI (2010).

Como dito anteriormente, os usuários são capazes de configurar totalmente e gerenciar os recursos de suas próprias fatias. A fim de conseguir isso, o NOC cria uma rede de gerenciamento privada para cada usuário. Esta rede é separada da Internet pública e pode ser acessada apenas por meio do Servidor de Acesso Virtual (VAS – *Virtual Access Server*) (passo 4 da Figura 3.3). Esta solução evita as configurações não autorizadas ou não programadas, e o VAS também atua como um mecanismo de autenticação de acesso.

FEDERICA oferece um conjunto de ferramentas automatizadas (chamado de *toolbench* (KRZYWANIA; DOLATA; SZEGEDI, 2010)) para apoiar os processos do NOC. Dentre as ferramentas, existe uma solução que realiza o gerenciamento de fatias para facilitar o gerenciamento dos recursos, acelerar o processo de configuração, e fazer o aprovisionamento de forma mais dinâmica. Esta ferramenta permite ao operador do NOC criar, adicionar recursos virtuais, e exportar fatias para os SPs. Após criar uma fatia, o NOC cria credenciais no VAS e define a data e a hora de sua expiração. O NOC também define o mapeamento da fatia entre as máquinas e os enlaces físicos correspondentes, e cria VLANs para concluir a criação da fatia (ESTEVEVES; GRANVILLE; BOUTABA, 2013).

O monitoramento de nós físicos na FEDERICA é realizado principalmente através do protocolo SNMP. Além disso, FEDERICA conta com a interface de linha de comando remota (RCLI – *Remote Command Line Interface*) do VMWare para monitorar os nós virtuais.

3.1.4 ProtoGENI

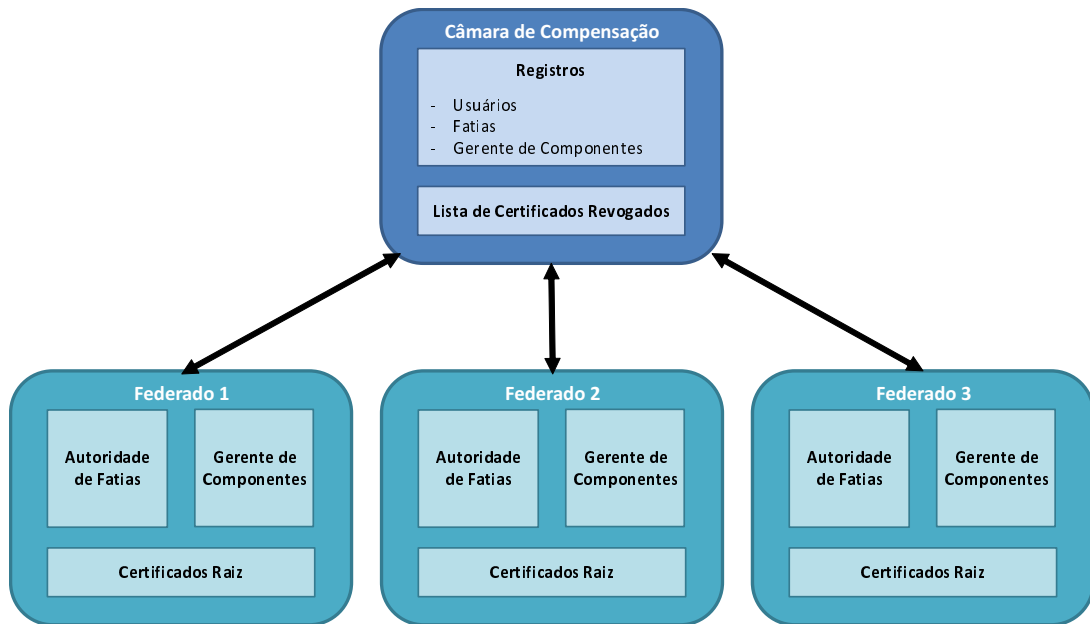
ProtoGENI (PROTOGENI, 2015) é um protótipo de implementação do *Global Environment for Network Innovations* (GENI). O ProtoGENI propõe e executa o *framework* de controle do GENI, incluindo o gerenciamento e alocação de recursos para experimentos autorizados e autenticados. Seus principais componentes e entidades de funcionamento são (LI; HONG; WITT, 2013):

- Câmara de Compensação (CH – *Clearinghouse*): centro de registro e autorização.
- Gerente de Componente (CM – *Component Manager*): provedor de serviços; gerencia recursos em um local particular.
- Fatia (*Slice*): *container* de recursos; proporciona um pedaço de um ambiente de teste de rede virtualizado para um experimento;
- Autoridade de Fatias (SA – *Slice Authority*): gerencia as fatias e autentica seus usuários.
- Lascas (*Slivers*): recursos de computação concedido a uma fatia.
- RSpec (*Resource Specification*): especificação de recurso; mecanismo usado para publicar, solicitar e descrever os recursos.
- Vnode (*Virtual node*): nó virtual; um nó na lasca atual, que compartilha os recursos no servidor físico com outras fatias usando a virtualização.
- VLAN: Ligações de VLAN para conectar, de outra forma, nós experimentais separados.

A Câmara de Compensação é o ponto central do gerenciamento no ProtoGENI, responsável pelo registro e acompanhamento de todas as fatias, usuários e Gerentes de Componentes, e permitindo a troca de certificados-raiz entre os membros do ProtoGENI. Autoridades de Fatias são os pontos de entrada para os usuários requisitarem fatias aos Gerentes de Componentes pertencentes aos participantes da federação ProtoGENI. Gerentes de Componentes controlam o provisionamento de recursos dentro de um membro da federação ProtoGENI. A Figura 3.4 ilustra a visão geral da arquitetura da federação ProtoGENI, apresentando seus principais componentes.

Para obter uma fatia, um usuário precisa registrá-la no nível de uma Autoridade de Fatias e obter uma credencial correspondente. A credencial permite criar lascas usando Gerentes de Componentes da federação ProtoGENI. Em seguida, o usuário contacta e solicita *tickets* aos Gerentes de Componentes. Os *tickets* são credenciais especiais que garantem que os recursos solicitados sejam vinculado a uma determinada fatia.

Figura 3.4 – Principais entidades do ProtoGENI



Fonte: ESTEVES; GRANVILLE; BOUTABA (2013).

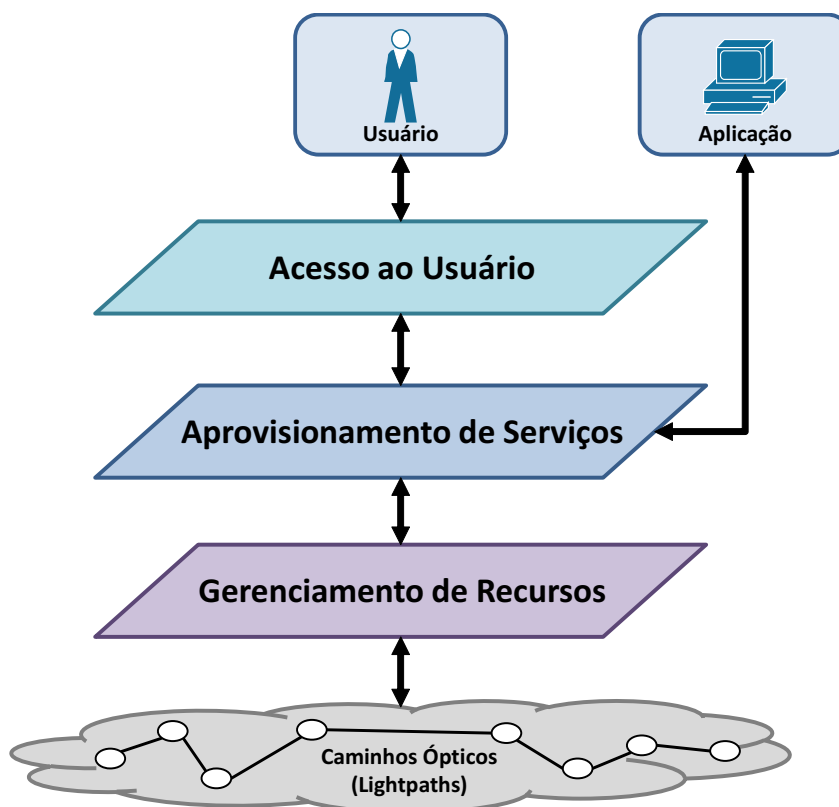
3.1.5 UCLP

UCLP (*User Controlled Lightpaths*) (WU et al., 2003; BOUTABA; GOLAB; IRAQI, 2004) é um sistema distribuído de gerenciamento e controle de redes ópticas entre múltiplos domínios. UCLP possibilita aos usuários finais tratar recursos de rede como objetos de software, permitindo-lhes aprovisionar e reconfigurar dinamicamente redes ópticas (na camada física) (CHOWDHURY; BOUTABA, 2010). Os usuários são capazes de criar, destruir, anunciar, alugar, agrupar ou dividir caminhos ópticos (*lightpaths*) dentro de um único domínio, ou em vários domínios de gerenciamento independentes, para criar redes IP lógicas personalizadas (ESTEVES; GRANVILLE; BOUTABA, 2013).

UCLP está estruturado em três camadas principais (BOUTABA; GOLAB; IRAQI, 2004): a camada de acesso do usuário (*user access layer*), a camada de aprovisionamento de serviços (*service provisioning layer*), e a camada de gerenciamento de recursos (*resource management layer*), conforme ilustrado na Figura 3.5.

A camada de acesso do usuário é o ponto de entrada a partir do qual os usuários humanos podem solicitar e gerenciar objetos do caminho óptico através de uma interface *Web*. Operações de caminhos ópticos são executadas por um conjunto de serviços definidos na camada de aprovisionamento, que também atua como um ponto de acesso para aplicações externas. A camada de gerenciamento de recursos é composta por um conjunto de agentes de recursos responsá-

Figura 3.5 – Estrutura do UCLP



Fonte: ESTEVES; GRANVILLE; BOUTABA (2013).

veis por se comunicar com dispositivos físicos de tecnologia específica (*e.g.*, SONET/SDH). O monitoramento na UCLP é realizada principalmente por meio do protocolo SNMP padrão.

O UCLPv1.4 (RECIO et al., 2005) introduziu o processo de descoberta de topologia dinâmico e permitiu o auto-roteamento através de algoritmos inteligentes, além da capacidade de configuração manual de *lightpaths* já disponível. Posteriormente, o UCLPv2 (GRASA et al., 2008) estendeu o UCLP com o uso de arquitetura orientada a serviços (SOA – *Service Oriented Architecture*) e tecnologias de *workflow* com o objetivo de formar a arquitetura de apoio para permitir a interligação de instrumentos, fatias de tempo, e sensores; e para a incorporação de roteadores e *switches* virtuais.

3.1.6 VNARMS

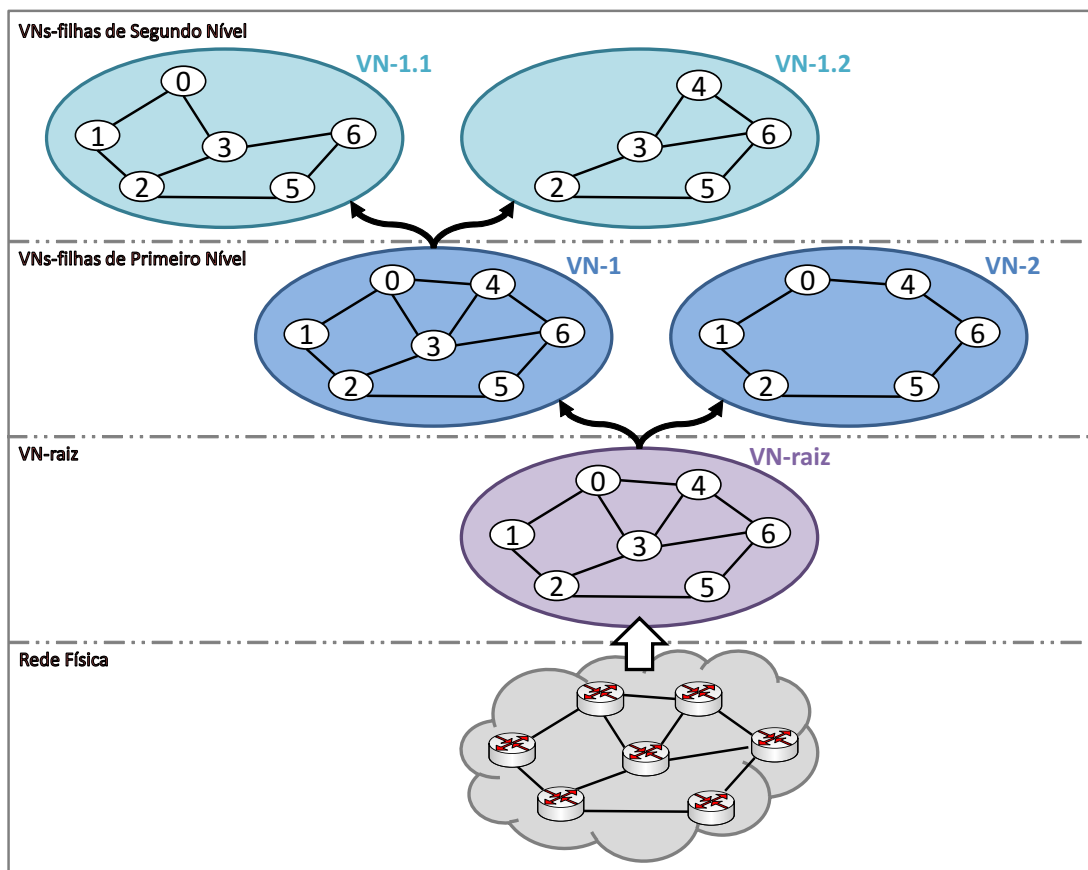
O *Virtual Network-based Autonomic network Resource control and Management System* (VNARMS) (KIM; LEON-GARCIA, 2007) baseia-se no gerenciamento autônomo para construir redes virtuais com garantias de desempenho.

Em cada rede virtual, há duas entidades básicas: o gerenciador de recursos de redes virtuais

(VNRM – *Virtual Network Resource Manager*) e o agente de recursos (RA – *Resource Agent*). O VNRM é responsável pelo gerenciamento da rede virtual (VN) por meio do controle de um conjunto de RAs distribuídos, que se comunicam com elementos de redes individuais. Ambas as entidades (*i.e.*, VNRM e RA) são autônomas, e monitoram os recursos gerenciados para identificar problemas e reagir de acordo com a necessidade.

O VNARMS usa o conceito de uma rede virtual raiz (VN-raiz) para abstrair a rede física e criar redes virtuais. Além disso, o VNARMS baseia-se nos serviços diferenciados (*DiffServ*) para garantir a qualidade de serviço (QoS – *Quality of Service*). A VN-raiz pode gerar múltiplas VNs filhas (VN-filha) que satisfazem o QoS definido para a(s) mesma(s). Quando um SP solicita uma nova rede virtual, o VNRM da VN-raiz calcula uma topologia com base nos requisitos do Acordo de Nível de Serviço (SLA – *Service Level Agreement*) do SP, gera uma VN-filha da VN-raiz, e instancia um novo VNRM para a VN-filha. O RA da VN-raiz também cria novas RAs para gerenciar os recursos virtuais individuais da VN-filha. Redes virtuais de segundo nível podem ser provisionadas, de uma forma recursiva, a partir de uma VN-filha previamente criada, como ilustrado na Figura 3.6.

Figura 3.6 – Aprovisionamento de VNs no VNARMS



Fonte: KIM; LEON-GARCIA (2007).

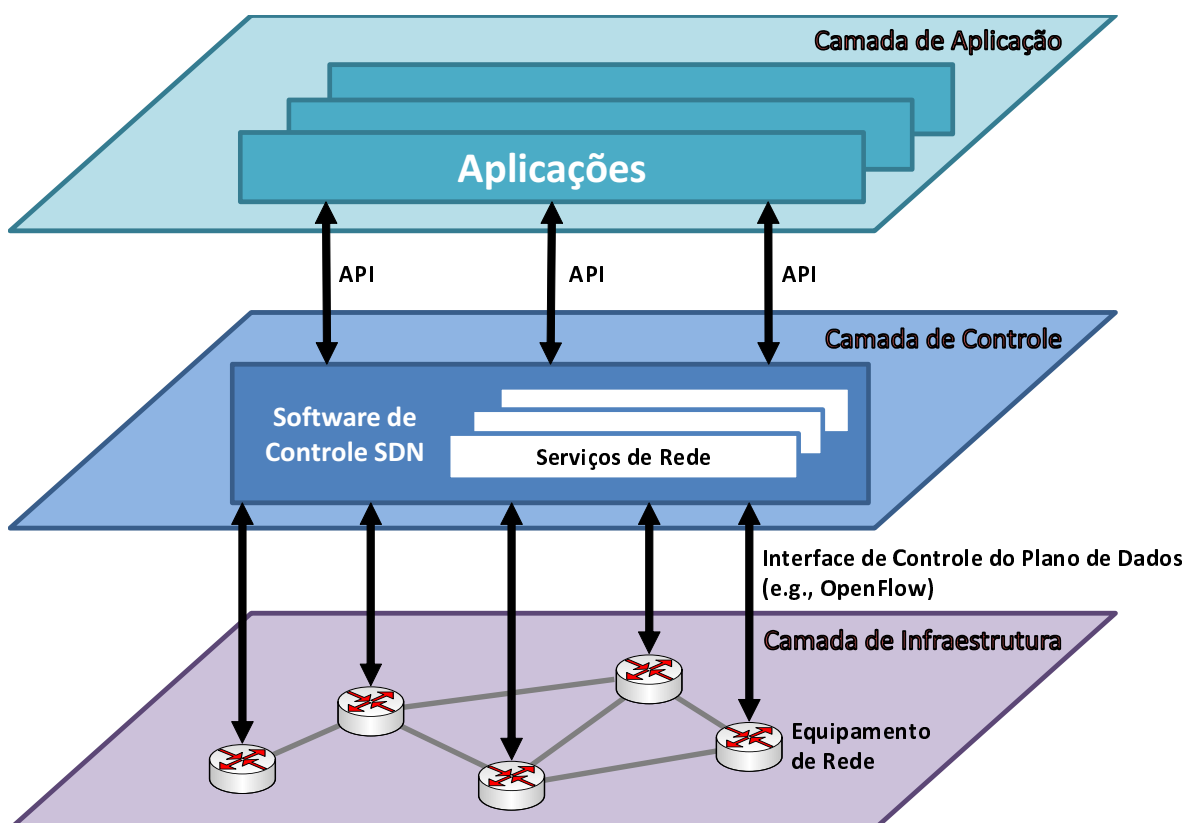
Na Figura 3.6, VN-1.1 e VN-1.2, redes virtuais de segundo nível, foram provisionadas a partir da VN-1. De modo análogo, VN-1 e VN-2, ambas VN-filha, foram provisionadas a partir da VN-raiz. A VN-raiz, por sua vez, é uma abstração da rede física.

3.1.7 OpenFlow

OpenFlow (MCKEOWN et al., 2008) é uma nova tecnologia baseada no conceito de redes definidas por software (SDN – *Software-defined Networking*). SDN é uma arquitetura de rede emergente onde o controle da rede é diretamente programável e dissociado do encaminhamento. O conceito de SDN consiste em romper com o modelo convencional de redes, separando o plano de controle do plano de dados, sendo que o plano de dados permanece nos equipamentos de rede e o plano de controle é movido para um controlador externo baseado em software, que mantém uma visão global da rede.

A arquitetura SDN, ilustrada na Figura 3.7, pode ser logicamente representada em três camadas: infraestrutura, controle e aplicação.

Figura 3.7 – Arquitetura SDN



Fonte: OPEN NETWORKING FOUNDATION (2012).

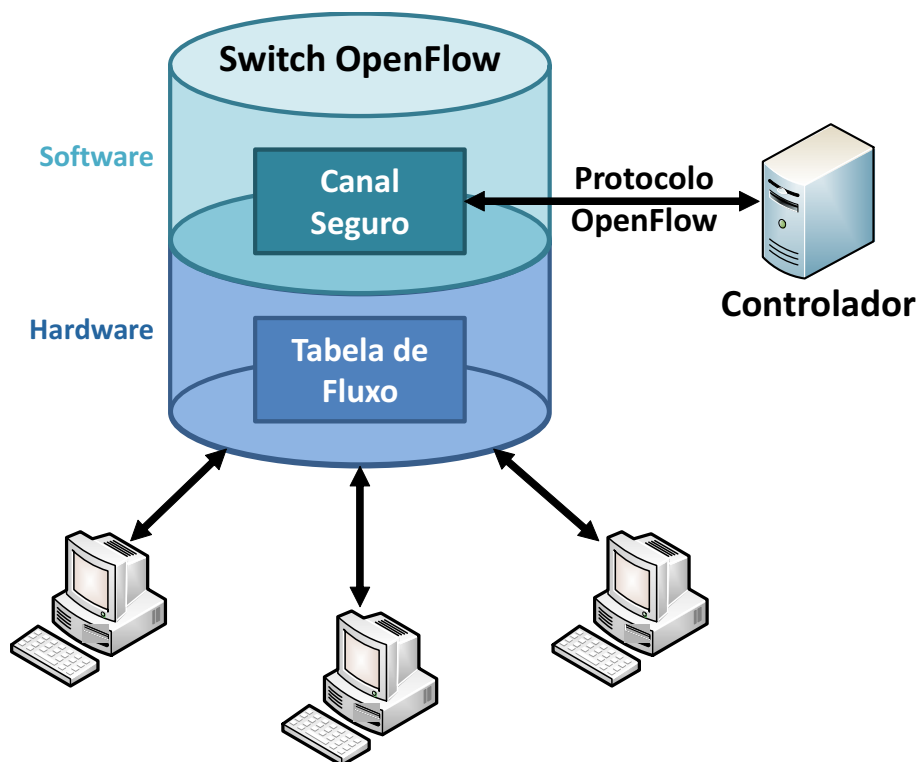
Na camada de infraestrutura encontra-se a rede física subjacente. Em SDN, os serviços de

rede são abstraídos da rede física, formando a camada de controle e permitindo, assim, que a rede seja tratada como uma entidade lógica ou virtual. A inteligência e o estado da rede são logicamente centralizados, e podem ser programados por aplicações da camada de aplicação, que usam as interfaces de programação de aplicações (API – *Application Programming Interface*) suportadas pelo *software* de controle SDN (OPEN NETWORKING FOUNDATION, 2012).

O OpenFlow define um protocolo aberto e padronizado para ser uma interface de comunicação entre o controlador e os equipamentos de rede. Para isso, ele é implementado em ambos os lados, permitindo ao controlador realizar a configuração e a manipulação das tabelas de encaminhamento existentes nos ativos de rede. O OpenFlow usa o conceito de fluxo para identificar o tráfego de rede, baseando-se em regras pré-definidas que podem ser programadas estática ou dinamicamente através do software de controle SDN (YAP et al., 2010).

Os equipamentos de rede que implementam o OpenFlow em sua arquitetura são denominados *switches* OpenFlow (ou OpenFlow Switches). Esses *switches* são normalmente gerenciados por um controlador centralizado, utilizado para criar, remover e modificar as entradas de fluxo. Um *switch* OpenFlow é composto por pelo menos três partes, como mostra a Figura 3.8: uma tabela de fluxo, um canal seguro e o protocolo OpenFlow.

Figura 3.8 – Arquitetura do Switch OpenFlow



Fonte: MUNTANER (2012).

A tabela de fluxo é a estrutura de dados que contém as regras de fluxos. As regras de fluxos

têm a função de comandar o *switch* para realizar algumas ações em certos fluxos por um tempo determinado, *e.g.*, encaminhar pacotes de um certo fluxo a uma determinada porta do *switch*. O canal seguro conecta o *switch* ao controlador remoto, permitindo que os comandos trafeguem de forma segura entre eles (MCKEOWN et al., 2008; MUNTANER, 2012; FERNANDEZ, 2013).

Para que um *switch* seja compartilhado com vários usuários ou aplicativos, o OpenFlow requer a adição de uma camada de virtualização. Para este fim, foi introduzida uma camada denominada FlowVisor (SHERWOOD et al., 2010), que permite que um *switch* OpenFlow seja devidamente fatiado e atribuído a diferentes usuários.

Um dos principais problemas com que o FlowVisor tem que lidar é o gerenciamento do isolamento entre as fatias, que é implementado através de uma série de mecanismos. Para o isolamento de banda, o FlowVisor configura filas de largura de banda mínimas para cada fatia que compartilha uma porta de um *switch*. Para lidar com o isolamento de CPU, o FlowVisor limita o número de mensagens de controle que um usuário pode enviar. Outros mecanismos de isolamento incluem a limitação do número de entradas por fatias nas tabelas de fluxo, e a reescrita de mensagens de controle originadas em uma fatia especial, para evitar conflitos com outras fatias.

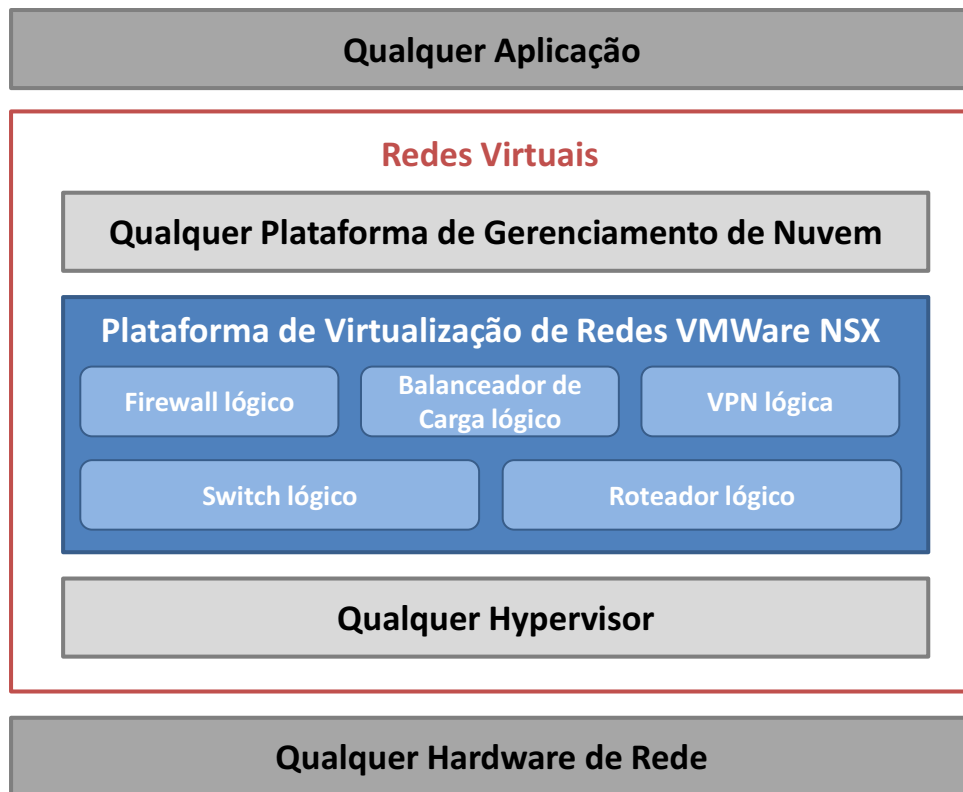
3.1.8 VMWare NSX

VMware NSX (VMWARE, 2015) é uma plataforma proprietária de virtualização de rede e segurança que fornece um serviço completo e programável de redes virtuais para máquinas virtuais, implantadas sobre qualquer *hardware* de rede IP.

O NSX reproduz todo o modelo de redes, da camada 2 à camada 7, em *software*, permitindo que diversas topologias de rede sejam provisionadas e gerenciadas independente do hardware subjacente. O NSX conta com um conjunto de elementos lógicos de redes e serviços que podem ser habilitados, tais como: *switches*, roteadores, *firewalls*, balanceadores de carga e VPN (*Virtual Private Network*). Além disso, o NSX dispõe de QoS, monitoramento e segurança, de modo que seus usuários possam criar redes virtuais isoladas através de combinações personalizadas dessas capacidades (VMWARE, 2013b).

A plataforma de virtualização de redes NSX pode ser implantada sobre qualquer *hardware* de rede, residir com qualquer *hypervisor*, conectar-se com qualquer plataforma de gerenciamento de nuvem (*e.g.*, vCloud, OpenStack, CloudStack, ou outra através de integração com a NSX API) e ser utilizada por quaisquer aplicações (VMWARE, 2013a), conforme esquematizado na Figura 3.9.

Figura 3.9 – Esquema de virtualização de redes do VMWare NSX



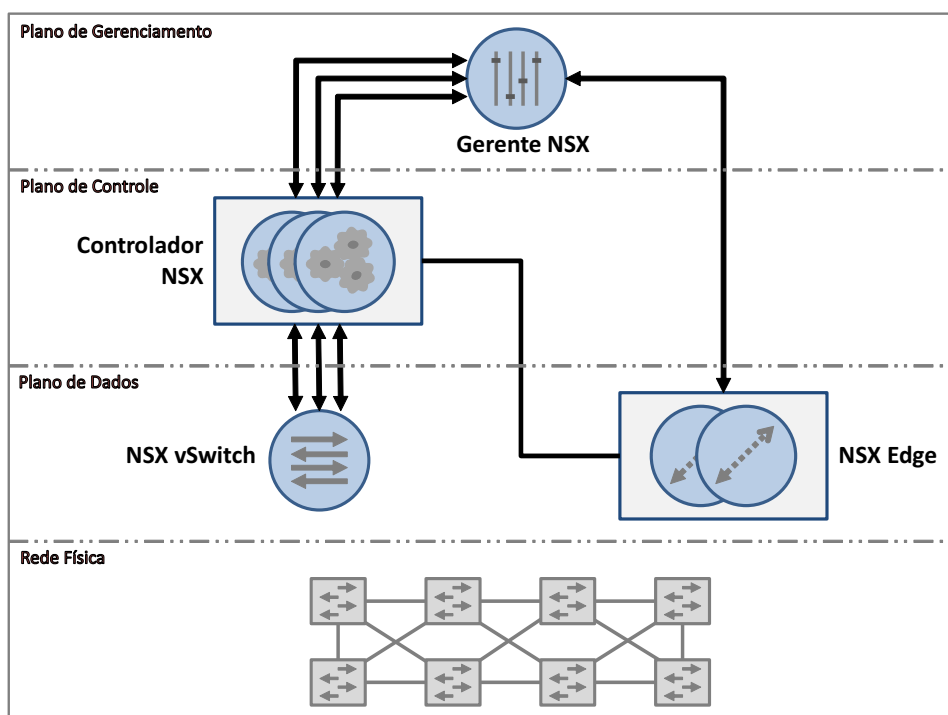
Fonte: VMWARE (2013b).

A solução de virtualização de redes VMWare NSX é estruturada em três planos: plano de gerenciamento, plano de controle e plano de dados. Além disso, sua plataforma é composta por quatro componentes básicos: Gerente NSX, Controlador NSX, NSX Edge e NSX vSwitch. A Figura 3.10 apresenta a interação entre os componentes.

O Gerente NSX fornece uma interface gráfica (GUI – *Graphical User Interface*) baseada na *Web* para a interação do usuário com a API do Controlador NSX, de modo a permitir a configuração e a administração do sistema, e ajudar na solução de problemas. Além disso, o administrador do sistema pode visualizar *logs* e *status* de conectividade de todos os componentes do VMware NSX e elementos de rede virtuais, além de poder tirar *snapshots* do estado da rede virtual para *backup*, restauração, e arquivo (NAGUIB, 2013).

O Controlador NSX, ou *Cluster Controlador*, é um sistema distribuído de alta disponibilidade responsável pelo gerenciamento dos módulos de comutação (*switching*) e roteamento nos *hypervisors* (VMWARE, 2013c). Mesmo podendo ser fisicamente distribuído, o *cluster controlador* fornece uma camada de controle logicamente centralizada. Além disso, ele aceita requisições de plataformas de gerenciamento superiores (*e.g.*, vCloud, OpenStack), calcula a topologia de rede virtual e, de forma proativa, programa os NSX vSwitches e os NSX Edges

Figura 3.10 – Interação entre componentes básicos do VMWare NSX



Fonte: SOFTWARE DEFINED CLOUD NETWORKING (2014).

(NAGUIB, 2013).

Os NSX Edgdes oferecem serviços de segurança de borda e de *gateway* de rede para isolar uma rede virtualizada. Um NSX Edge pode ser instalado como um roteador lógico (distribuído) ou como um gateway de serviços. O roteador lógico oferece roteamento distribuído com o espaço de endereço IP da rede inquilina e isolamento de caminho de dados. Assim, as máquinas virtuais e as cargas de trabalho que residem no mesmo *host* em diferentes sub-redes podem se comunicar umas com as outras, sem precisar atravessar uma interface de roteamento tradicional. O *gateway* conecta redes isoladas (*stub network*) a redes compartilhadas, fornecendo serviços comuns de *gateway*, tais como: DHCP (*Dynamic Host Configuration Protocol*), VPN, NAT (*Network Address Translation*), roteamento dinâmico, e balanceamento de carga.

Os NSX vSwitches são *switches* com *hypervisor*, que possuem um plano de dados (de camadas 2 a 4) programável e um banco de dados de configuração. O Controlador NSX programa cada vSwitch para que a topologia da rede virtual corresponda com aquela que as máquinas virtuais estão associadas.

O VMWare NSX é uma plataforma extensível, permitindo que outros registrem seus serviços com o Controlador NSX e inseriram as respectivas capacidades nas redes virtuais. Além disso, o VMWare NSX usa interfaces e protocolos abertos, o que possibilita que ecossistemas de terceiros se integrem de modo mais fácil com o VMware NSX (NAGUIB, 2013).

3.2 Classificação das Soluções de Gerenciamento

Nesta seção serão classificados os projetos de gerenciamento de NVEs em termos de alvos, funções e abordagens de gerenciamento, que combinados proporcionam uma compreensão holística de como é, atualmente, o gerenciamento os NVEs modernos. Estes critérios são comumente usados para organizar os problemas de gerenciamento de redes em geral, mas também são aplicáveis a NVEs (ESTEVEZ; GRANVILLE; BOUTABA, 2013).

3.2.1 Alvos de gerenciamento

Um alvo de gerenciamento refere-se a um componente gerenciado de um NVE. Componentes gerenciados podem pertencer a diferentes camadas de um NVE (*i.e.*, física, virtual, ou aplicação). Alvos individuais podem ser combinados em outros mais complexos (*e.g.*, redes virtuais), demandando esforços de gerenciamento adicionais. Aqui, os alvos de gerenciamento são classificados em gerenciamentos de nós, de enlaces, e de redes.

- **Gerenciamento de nós** – lida com a operação de nós virtuais e físicos de um NVE, incluindo a criação inicial de nós virtuais sobre o substrato e a migração de nós.
- **Gerenciamento de enlaces** – aborda aspectos específicos relacionados com a configuração e a operação de enlaces físicos e virtuais, tais como isolamento de enlace virtual e escalonamento de fluxo.
- **Gerenciamento de redes** – engloba não apenas um único nó ou enlace do NVE, mas uma rede virtual inteira, incluindo redes virtuais que abrangem várias redes físicas.

A estruturação das atividades de gerenciamento de acordo com o seu alvo pode ajudar os operadores de NVE a efetivamente identificar e delegar tarefas de gerenciamento (*e.g.*, fornecendo isolamento entre os vários enlaces virtuais) com base em seu alvo.

3.2.2 Funções de gerenciamento

Para discutir como o gerenciamento de redes foi abordado por projetos de virtualização de redes, foram identificadas aqui as principais funções de gerenciamento que são essenciais em qualquer solução de gerenciamento de NVEs. São elas:

- **Aprovisionamento de recursos** – no contexto da virtualização de redes, consiste em definir o mapeamento dos recursos de redes virtuais (*e.g.*, nós, *links*) para suas contrapartes físicas e conceder aos operadores de SPs acesso às suas redes virtuais.

- **Monitoramento** – envolve a coleta do estado atualizado dos recursos físicos e das suas redes virtuais associadas. Filtragem, correlação, agregação e compactação de informações de monitoramento de diferentes fontes são necessárias para reduzir o *overhead* de gerenciamento.
- **Interfaceamento** – interfaces são necessárias para que os InPs e os SPs possam acessar, operar, manter e administrar nós e enlaces físicos e virtuais. Dispositivos de redes físicos devem apresentar uma interface de gerenciamento uniforme, que permita que nós e enlaces virtuais localizados em nós e enlaces físicos heterogêneos façam parte da mesma rede virtual e sejam facilmente gerenciáveis pelo SP. Dentre as diversas funcionalidades que uma interface de gerenciamento pode suportar, são desejáveis: recuperação de variáveis de estado; suporte à notificação; configuração de atributos individuais de recursos virtuais como capacidade de processamento e de memória dos nós virtuais, largura de banda dos enlaces virtuais e tabelas de roteamento; e operações de gerenciamento de nós e enlaces virtuais como registro, criação, remoção, cópia, inicialização, desligamento e migração.

Estas funções (*i.e.*, provisionamento, monitoramento e interfaceamento), fundamentais para as redes tradicionais, ganham mais importância quando se trata de NVEs. O provisionamento permite que os SPs instanciem redes virtuais e as utilizem. O monitoramento dá suporte a várias outras tarefas de gerenciamento, como por exemplo o gerenciamento de falhas e faturamento. O interfaceamento define como os aplicativos de gerenciamento se comunicam com os recursos do NVE e permite a interoperabilidade.

3.2.3 Abordagens de gerenciamento

Soluções de gerenciamento de NVEs variam na forma como os gerentes e agentes são organizados. Algumas soluções dependem de um nó centralizado responsável por realizar todas as tarefas de gerenciamento, enquanto outros sistemas permitem que múltiplos nós distribuídos compartilhem a tarefa de gerenciar a infraestrutura. Sistemas de gerenciamento empregados nos NVEs também podem ter diferentes níveis de automação. Gerenciamento autônomo ajuda a reduzir a intervenção humana, e permite adaptação dinâmica a mudanças na rede. Gerenciamento baseado em políticas auxilia os administradores de InP a lidar com a complexidade inerente de um NVE e a automatizar a configuração de recursos de acordo com os objetivos de negócios de alto nível. Os 3 tipos de abordagens de gerenciamento para fins de classificação são:

- **Gerenciamento centralizado** – uma única estação de gerenciamento localizada no InP é responsável por supervisionar o gerenciamento dos recursos de rede do InP. Idem se

localizada no SP.

- **Gerenciamento distribuído** – múltiplos nós trabalham de forma cooperativa para realizar as tarefas de gerenciamento.
- **Gerenciamento autônomo e baseado em políticas** – gerenciamento autônomo permite ao NVE gerenciar a si mesmo de acordo com o estado atual da rede. As soluções de gerenciamento autônomo normalmente contam com políticas de alto nível que são regras gerais definidas para reger o funcionamento dos dispositivos de redes subjacentes. Em NVEs, as políticas também são usadas pelos InPs para reforçar o isolamento entre as redes virtuais, controlando as permissões de acesso para cada SP.

3.2.4 Classificação

Compreender tais abordagens de gerenciamento pode ajudar os administradores de NVEs a avaliar o compromisso entre o tamanho do NVE e a complexidade da solução necessária para gerenciá-lo.

A Tabela 3.1 mostra os alvos de gerenciamento suportados por cada projeto.

Tabela 3.1 – Classificação quanto aos alvos de gerenciamento

Projetos	Alvos de Gerenciamento		
	Nós	Enlaces	Redes
4WARD	✓	✓	✓
AUTOI	✓	✓	✓
FEDERICA	✓	✓	✓
ProtoGENI	✓	✓	✓
UCLP		✓	✓
VNARMS	✓	✓	✓
OpenFlow	✓		
VMWare NSX	✓	✓	✓

Fonte: do autor (2015).

As Tabelas 3.2 e 3.3 mostram, respectivamente, as funções e as abordagens de gerenciamento suportadas por cada projeto.

Comparando-se as propostas pesquisadas (Tabelas 3.1, 3.2 e 3.3), verificou-se que 4WARD, AUTOI e VMWare NSX cobrem a maior parte dos critérios identificados, reflexo dos esforços desses projetos em considerar o gerenciamento na fase de concepção. As outras propostas representam avanços significativos na área e enfatizam a tendência em considerar o gerenciamento

Tabela 3.2 – Classificação quanto às funções de gerenciamento

Projetos	Funções de Gerenciamento		
	Aprovisionamento	Monitoramento	Interfaceamento
4WARD	✓	✓	✓
AUTOI	✓	✓	✓
FEDERICA	✓	✓	✓
ProtoGENI	✓	✓	✓
UCLP	✓	✓	✓
VNARMS	✓		
OpenFlow	✓		
VMWare NSX	✓	✓	✓

Fonte: do autor (2015).

Tabela 3.3 – Classificação quanto à abordagem de gerenciamento

Projetos	Abordagens de Gerenciamento		
	Centralizado	Distribuído	Autônomo
4WARD		✓	✓
AUTOI		✓	✓
FEDERICA	✓		
ProtoGENI	✓		
UCLP		✓	
VNARMS		✓	✓
OpenFlow	✓		
VMWare NSX	✓		✓

Fonte: do autor (2015).

como um requisito primordial em projetos de redes futuros.

Com relação às diferentes perspectivas analisadas, notou-se que alguns aspectos de gerenciamento receberam ou estão recebendo mais atenção do que outros. Por exemplo, os três alvos de gerenciamentos (*i.e.*, nós, enlaces e redes) foram abordados por quase todas as soluções de gerenciamento, exceto pelos projetos UCLP e OpenFlow. Além disso, as três funções de gerenciamento (*i.e.*, provisionamento, monitoramento e interfaceamento) também foram implementadas por quase todos os protótipos apresentados, com exceção do VNARMS e do OpenFlow, que não implementaram monitoramento nem interfaceamento.

Outro fato interessante é que os gerenciamentos distribuído e autônomo foram considerados na metade das soluções estudadas, o que pode refletir, talvez, uma mudança de paradigma no projeto tradicional de gerenciamento de redes, mesmo que algumas funções (*e.g.*, de registro)

ainda sejam realizadas por entidades centralizadas.

3.3 Gerenciamento de NVEs em ambientes heterogêneos

O gerenciamento de NVEs construídos sobre substratos físicos heterogêneos (*i.e.*, com equipamentos e tecnologias diferentes) é complicado, pois cada fabricante geralmente tem sua própria solução de gerenciamento. Sendo assim, várias ferramentas independentes são necessárias para realizar uma tarefa de gerenciamento. Para minimizar a dificuldade de gerenciar um conjunto diversificado de recursos físicos, interfaces de gerenciamento padronizadas devem ser definidas de modo a permitir a interoperabilidade das diferentes plataformas de virtualização com as ferramentas de gerenciamento.

Em se tratando de gerenciamento em ambientes heterogêneos, interoperabilidade e padronização são requisitos primordiais. Sendo assim, o SNMP é a escolha mais natural para compor uma interface de gerenciamento, tendo em vista ser suportado por quase todos os modelos e marcas de equipamentos de rede. Contudo, para a implementação de uma interface de gerenciamento de NVE baseadas em SNMP, há necessidade de MIBs específicas que suportem estes requisitos.

Uma abordagem de MIB para roteadores virtuais, conhecida como VR-MIB (*Virtual Router - Management Information Base*), foi parcialmente implementada pelo IETF (STELZER et al., 2005). Esta MIB foi desenvolvida no contexto das VPNs de camada 3 e é composta de objetos relacionados à configuração em alto-nível de VRs, além de objetos para a coleta de estatísticas sobre os dispositivos, conforme descrito detalhadamente na Seção 3.3.1.

Daitx, Esteves e Granville (2011) propuseram uma interface de gerenciamento de NVEs baseada no SNMP, utilizando a VR-MIB. Além disso, propuseram uma extensão do módulo VR-MIB para flexibilizar a vinculação entre as interfaces físicas e virtuais dos VRs. Para avaliar o desempenho da interface proposta, eles usaram algumas operações básicas de gerenciamento em duas plataformas de virtualização, XenServer e VMWare, e mostraram que o desempenho do SNMP depende amplamente da plataforma utilizada.

A principal desvantagem da VR-MIB é que ela não progrediu no caminho da padronização e permanece sem atualização desde 2006, fato este que desencoraja sua utilização em projetos atuais. Atualmente, não se conhece MIB que substitua suas funcionalidades no que diz respeito à configuração de roteadores virtuais, ficando a área sem solução baseada no SNMP. Sendo assim, a solução mais rápida para este problema é procurar uma MIB atual que possa absorver as funcionalidades da VR-MIB.

Uma abordagem de MIB para máquinas virtuais controladas por um *hypervisor*, conhecida como VMM-MIB (*Virtual Machine Manager - Management Information Base*), vem sendo implementada pelo IETF (ASAI et al., 2014). Um roteador virtual poder ser implementado em uma máquina virtual através da instalação de um roteador baseado em *software*, como por exemplo o Vyatta (BROCADE, 2015). Esse fato torna a VMM-MIB uma excelente candidata para absorver todas as funcionalidades da VR-MIB e possibilitar, assim, o gerenciamento pleno de VRs.

A VMM-MIB, detalhada na Seção 3.3.2, tem como funcionalidade servir de base de consulta sobre informações de gerenciamento de VMs (*Virtual Machines*) e *hypervisors* e, por isso, a grande maioria dos seus objetos tem permissão somente-leitura (*read-only*) em sua definição. Sendo assim, para que esse módulo MIB suporte as operações de configuração, como por exemplo, criação e remoção de VR, objetos configuráveis são absolutamente necessários.

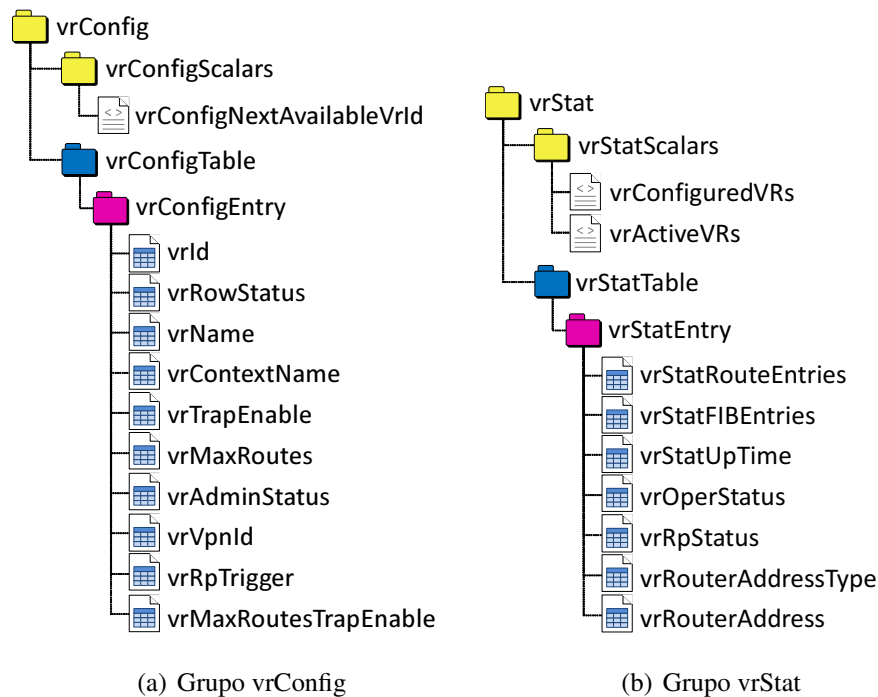
3.3.1 VR-MIB

A VR-MIB foi uma das primeiras iniciativas de gerenciamento que visavam redes com roteadores virtuais. Originalmente proposta pelo grupo de trabalho L3VPN do IETF (IETF, 2015) para a gerência de VPNs de camada 3, a VR-MIB teve sua proposta inicial em 2001, mas foi descontinuada em janeiro de 2006.

A VR-MIB mais atual, L3VPN-VR-MIB versão 4 (STELZER et al., 2005), define objetos para possibilitar a configuração em alto-nível de VRs (*e.g.*, criação e remoção de VRs), a coleta de estatísticas, e a associação de interfaces de rede virtuais com as físicas. Esses objetos estão organizados em 3 grupos de informações definidos como: `vrConfig`, `vrStat` e `vrIfConfig`.

O grupo `vrConfig`, mostrado na Figura 3.11(a), contém informações sobre indexação, criação e exclusão de VRs. Para ajudar a estação de gerenciamento na atribuição de um novo VR sem conflito, o próximo identificador de VR disponível pode ser recuperado acessando o objeto escalar `vrConfigNextAvailableVrId` do dispositivo gerenciado. Cada configuração de roteador virtual é armazenada em uma linha da tabela `vrConfigTable`, que contém um identificador único (`vrId`), o nome do VR (`vrName`) e o status da linha (`vrRowStatus`), sendo este último usado para criar e excluir os roteadores virtuais. Além disso, essa tabela contém informações sobre configurações internas de cada dispositivo virtual, como o nome do contexto SNMPv3 ou da comunidade SNMPv2c do VR (`vrContextName`), o identificador de VPN (`vrVpnId`), o número máximo de rotas virtuais suportadas (`vrMaxRoutes`) e um gatilho para ligar ou desligar os protocolos de roteamento (`vrRpTrigger`).

Figura 3.11 – Grupos de informações da VR-MIB



Fonte: STELZER et al. (2005).

O grupo `vrStat`, mostrado na Figura 3.11(b), contém 2 objetos escalares (`vrConfiguredVRs` e `vrActiveVRs`) com informações globais de dispositivos, como o número de VRs configurados no elemento de rede² (`vrConfiguredVRs`) e o número de VRs ativos no elemento de rede (`vrActiveVRs`). Além disso, este grupo contém uma tabela chamada `vrStatTable`, que possui os status administrativo e operacional de cada VR. Essa tabela contém diversas estatísticas como: o número atual de entradas de rota (`vrStatRouteEntries`), o número de entradas na FIB (*Forwarding Information Base*) (`vrStatFIBEntries`), o tempo decorrido após o VR entrar em operação (`vrStatUpTime`), o status operacional do VR (`vrOperStatus`), o tipo e o endereço IP de uma das interfaces do VR (`vrRouterAddressType` e `vrRouterAddress`).

²Elemento de rede, de acordo com a RFC 2216 (SHENKER; WROCLAWSKI, 1997), é qualquer componente de uma rede interna que manipula diretamente os pacotes de dados, *e.g.*, um roteador.

O grupo `vrIfConfig`, mostrado na Figura 3.11(c), é composto por uma tabela usada para a configuração de interfaces de VRs (`vrIfConfigTable`). Esta tabela contém todas as associações entre os roteadores virtuais e as interfaces de rede físicas, pois possui como índices da `vrIfConfigEntry` os identificadores `vrId` e `vrIfId`. Assim como na `vrConfigTable`, interfaces de roteadores virtuais podem ser criadas, excluídas ou modificadas editando-se a variável `vrIfConfigRowStatus`. Este objeto é usado para acionar as operações de criação e remoção ou para definir o estado de operação de cada uma das interfaces virtuais de VRs. A fim de definir uma nova interface virtual, o agente SNMP primeiro verifica se o VR está disponível e, em seguida, verifica se a interface de rede escolhida está disponível.

As tabelas `vrConfigTable` e `vrIfConfigTable` têm que suportar a criação de VRs e interfaces de rede virtuais, respectivamente, portanto seus objetos possuem permissão do tipo *read-create*. A tabela `vrStatTable` e as variáveis escalares armazenam somente informações estatísticas e administrativas para consulta, portanto seus objetos são todos do tipo *read-only*.

Além dos três grupos supracitados, o grupo `vrNotificationsPrefix` (Figura 3.11(d)) permite o envio de determinadas notificações (*traps*) ao gerente. Essa funcionalidade pode ser ativada ou desativada, para cada VR, manipulando-se a variável `vrTrapEnable` da tabela `vrConfigTable`. O grupo `vrNotificationsPrefix` é formado pelas *traps*: `vrUp`, `vrDown` e `vrMaxRoutesExceeded`. A notificação `vrUp` é gerada quando um VR está prestes a ser inicializado ou seu estado operacional (`vrOperStatus`) é alterado de desligado (*down*) para ligado (*up*). Analogamente, a notificação `vrDown` é gerada quando um VR está prestes a ser desligado ou seu estado operacional (`vrOperStatus`) é alterado de ligado (*up*) para desligado (*down*). A notificação `vrMaxRoutesExceeded` é gerada quando em um determinado VR o número de rotas (`vrStatRouteEntries`) excede o limite máximo definido em `vrMaxRoutes`. Esta última notificação (`vrMaxRoutesExceeded`) pode ser ativada ou desativada, para cada VR, manipulando-se a variável `vrMaxRoutesTrapEnable` da tabela `vrConfigTable`.

3.3.2 VMM-MIB

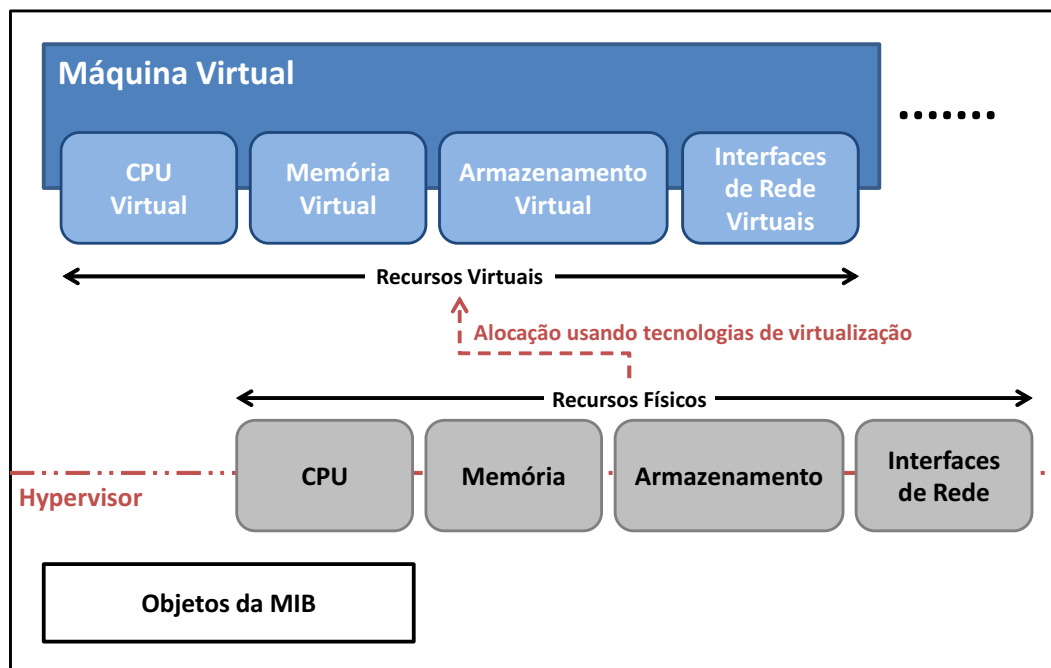
A VMM-MIB foi proposta no IETF, em julho de 2012, para o gerenciamento de Máquinas Virtuais (VMs – *Virtual Machines*) controladas por um *hypervisor* (ASAI et al., 2014). Um *hypervisor*, também conhecido como monitor de máquina virtual, controla várias máquinas virtuais em uma única máquina física, alocando recursos para cada VM usando tecnologias de virtualização. Portanto, esse módulo MIB contém informações sobre as máquinas virtuais e seus recursos (controlados por um *hypervisor*), bem como contém informações de *hardware* e

software relacionadas ao *hypervisor*.

O projeto da VMM-MIB foi derivado de MIBs específicas de empresas, como Xen e VMWare, e de um módulo MIB para interface de programação *libvirt*, no intuito de englobar as principais soluções de virtualização do mercado. No entanto, esta MIB tenta generalizar os objetos gerenciados para suportar outros *hypervisors*.

A VMM-MIB possui informações relativas ao sistema de um *hypervisor*, a lista de máquinas virtuais controladas, e recursos virtuais alocados para VMs. Sobre os recursos alocados, são especificados quatro tipos de recursos virtuais comuns a todos os *hypervisors*: CPUs (processadores), memória, dispositivos de armazenamento, e interfaces de rede. Um *hypervisor* aloca recursos virtuais às VMs a partir dos recursos físicos existentes, conforme esquematizado na Figura 3.12.

Figura 3.12 – Alocação de recursos virtuais

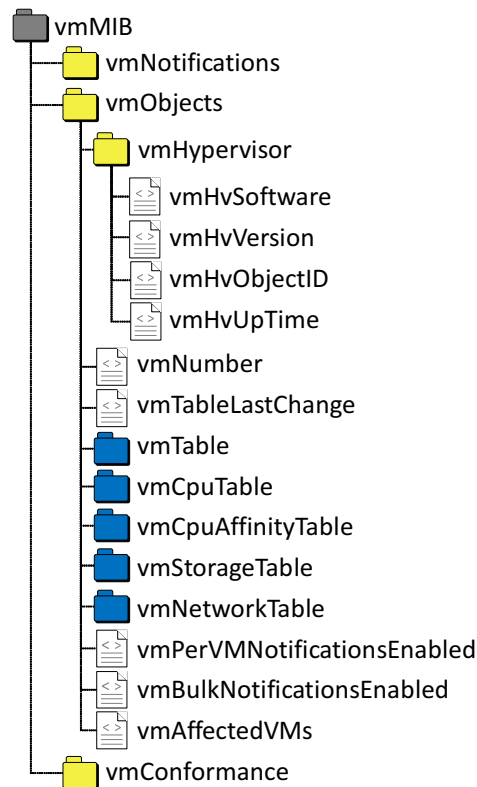


Fonte: ASAI et al. (2014).

A VMM-MIB está estruturada conforme a Figura 3.13, sendo composta de 5 tabelas e 9 objetos escalares, dos quais 4 escalares estão organizados no grupo `vmHypervisor`. Os objetos deste grupo fornecem informações básicas sobre o *hypervisor*, como a descrição do *software* (`vmHvSoftware`), a versão (`vmHvVersion`), o identificador (`vmHvObjectID`), e o tempo desde que o *hypervisor* foi reinicializado (`vmHvUpTime`). Os demais objetos escalares são: `vmNumber`, que contém o número de VMs existentes no *hypervisor*; `vmTableLastChange`, que é o valor de `vmHvUpTime` no momento da última criação ou exclusão de uma entrada na

tabela `vmTable`; `vmPerVMNotificationsEnabled`, que habilita a geração de notificações por VMs; `vmBulkNotificationsEnabled`, que habilita a geração de notificações por agrupamentos de VMs; e `vmAffectedVMs`, que contém a lista de VMs que tiveram seu estado alterado.

Figura 3.13 – Estrutura da VMM-MIB



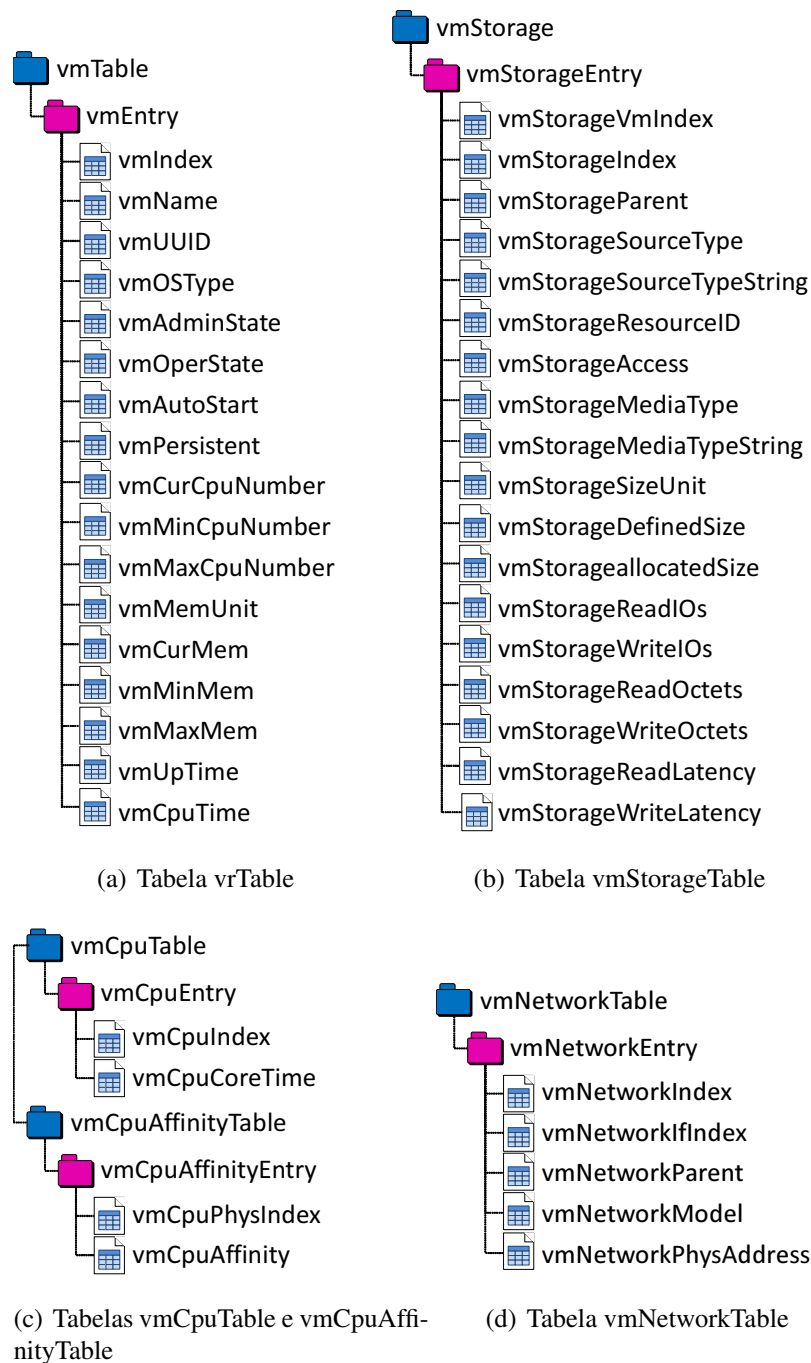
Fonte: ASAI et al. (2014).

Em relação às tabelas, temos que: a `vmTable` (Figura 3.14(a)), contém as VMs que são conhecidas pelo *hypervisor*; a `vmStorageTable` (Figura 3.14(b)), possui a lista de dispositivos de armazenamento virtual e seu mapeamento com as máquinas virtuais; a `vmCpuTable` (Figura 3.14(c)), faz o mapeamento de CPUs virtuais para máquinas virtuais; a `vmCpuAffinityTable` (Figura 3.14(c)), fornece a afinidade³ de cada processador virtual com uma CPU física; e a `vmNetworkTable` (Figura 3.14(d)), possui a lista de interfaces de rede virtuais e seu mapeamento para máquinas virtuais. As tabelas `vmTable` e `vmNetworkTable`, por serem mais relevantes para este trabalho, serão descritas com mais detalhes.

A tabela `vmTable`, mostrada na Figura 3.14(a), contém diversas informações sobre as VMs, como um identificador único (`vmIndex`), o nome (`vmName`), o identificador único uni-

³A especificação da afinidade restringe, em uma determinada VM, a atribuição de seus processadores virtuais a um subconjunto dos processadores físicos disponíveis em um sistema com múltiplos processadores.

Figura 3.14 – Tabelas da VMM-MIB



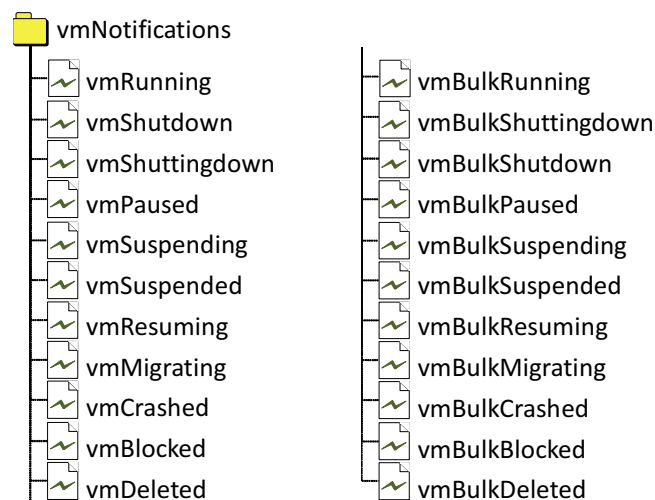
Fonte: ASAI et al. (2014).

versal (UUID – *Universally Unique Identifier*), o sistema operacional (`vmOSType`), os estados administrativo e operacional (`vmAdminState` e `vmOperState`), as configurações de auto-inicialização e de persistência (`vmAutoStart` e `vmPersistent`), o número atual, mínimo e máximo de CPUs atribuídas (`vmCurCpuNumber`, `vmMinCpuNumber` e `vmMaxCpuNumber`), o tamanho atual, mínimo e máximo de memória alocada (`vmCurMem`, `vmMinMem` e `vmMaxMem`), o tempo total que a VM está ativa (`vmUpTime`), e o tempo de uso da CPU (`vmCpuTime`).

A tabela `vmNetworkTable`, mostrada na Figura 3.14(d), contém informações de interfaces de rede virtuais, como um identificador único (`vmNetworkIndex`), um ponteiro para a `ifTable` da IF-MIB (MCCLOGHRIE; KASTENHOLZ, 2000) para o caso da interface estar também representada nesta outra MIB (`vmNetworkIfIndex`), outro ponteiro para a `ifTable` da IF-MIB no caso de haver uma interface de rede física pai correspondente (`vmNetworkParent`), o modelo de interface emulado pela interface de rede virtual (`vmNetworkModel`), e o endereço físico (MAC Address) (`vmNetworkPhysAddress`).

Eventos que causam transições entre os principais estados operacionais geram notificações (*traps*). A VMM-MIB define dois tipos de notificações: as notificações individuais (*per-VM*) e as notificações em massa (*bulk*). As notificações do tipo *per-VM* levam informações mais detalhadas, contudo a escalabilidade pode ser um problema. Já o mecanismo de notificação em massa tem o objetivo de reduzir o número de notificações que são capturadas por um gerente SNMP. As notificações *per-VM*, definidas pelos objetos listados na Figura 3.15(a), são habilitadas se `vmPerVMNotificationsEnabled` for verdadeiro. De modo análogo, as notificações em massa, definidas pelos objetos da Figura 3.15(b), são habilitadas se `vmBulkNotificationsEnabled` for verdadeiro.

Figura 3.15 – Objetos de notificação da VMM-MIB



(a) Notificações *per-VM*

(b) Notificações em massa

Fonte: ASAI et al. (2014).

Neste capítulo foram apresentados diversos trabalhos relacionados ao gerenciamento de NVEs e discutidas as dificuldades para gerenciar NVEs construídos sobre substratos físicos heterogêneos. Como resultado, evidenciou-se a importância de definir e avaliar uma interface de gerenciamento padronizada, que possibilite a interoperabilidade das diferentes plataformas

de virtualização com as ferramentas de gerenciamento, e que seja capaz de gerenciar de forma eficiente e escalável roteadores físicos que hospedam roteadores virtuais. No capítulo seguinte serão propostas cinco interfaces de gerenciamento baseadas em protocolos conhecidos, capazes de realizar inicialmente três operações básicas de gerenciamento de VRs: criação, recuperação de informação e remoção.

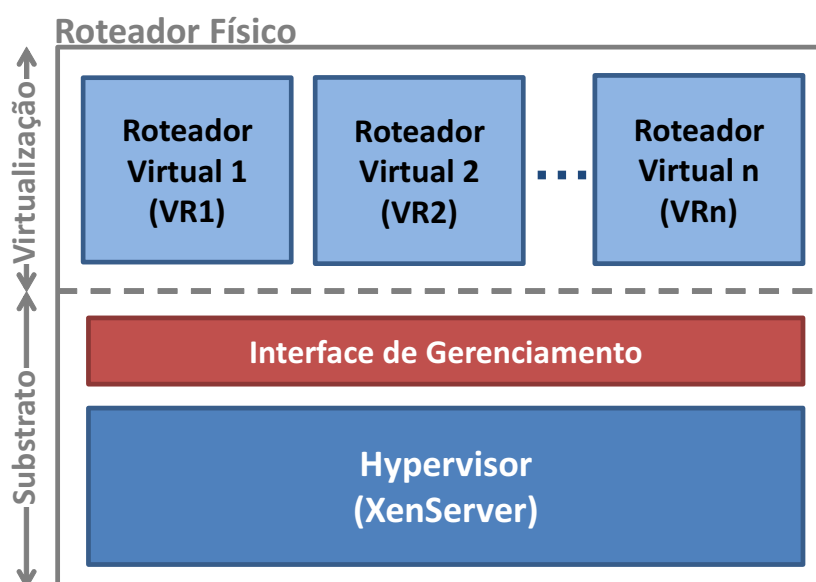
4 INTERFACES DE GERENCIAMENTO PARA VIRTUALIZAÇÃO DE ROTEADORES

Neste capítulo, primeiramente será descrita a arquitetura conceitual de um roteador físico que hospeda roteadores virtuais. Em seguida, sobre as interfaces de gerenciamento baseadas nos protocolos SNMPv2c, SNMPv3, NETCONF, RWS, e RWS sobre HTTPS, serão apresentados o modelo de dados e o fluxo de mensagens. Por fim, as interfaces propostas serão classificadas de acordo com os critérios apresentados no Capítulo 3 e situadas no panorama dos projetos supracitados.

4.1 Arquitetura

Um roteador físico é conceitualmente dividido em duas camadas: uma camada de substrato e uma camada de virtualização. A camada de substrato consiste no próprio dispositivo físico e contém uma camada de software, chamada *hypervisor*, que permite a implementação de roteadores virtuais (VRs) sobre ela. A camada de virtualização contém vários VRs isolados, que executam seus próprios planos de controle. Na arquitetura proposta, representada na Figura 4.1, a interface de gerenciamento está localizada entre o *hypervisor* e os VRs.

Figura 4.1 – Arquitetura de um roteador físico que hospeda VRs



Fonte: do autor (2015).

A interface de gerenciamento permite ao operador do provedor de infraestrutura (InP) acessar, utilizar, manter e administrar um roteador físico com suporte a virtualização (ESTEVEZ;

GRANVILLE; BOUTABA, 2013). No caso deste trabalho, o operador do InP usará a interface de gerenciamento para executar operações básicas de gerenciamento de VRs, como criação, remoção e recuperação de informações. A definição de um modelo de dados adequado é fundamental para suportar essas operações.

4.2 Modelo de Dados

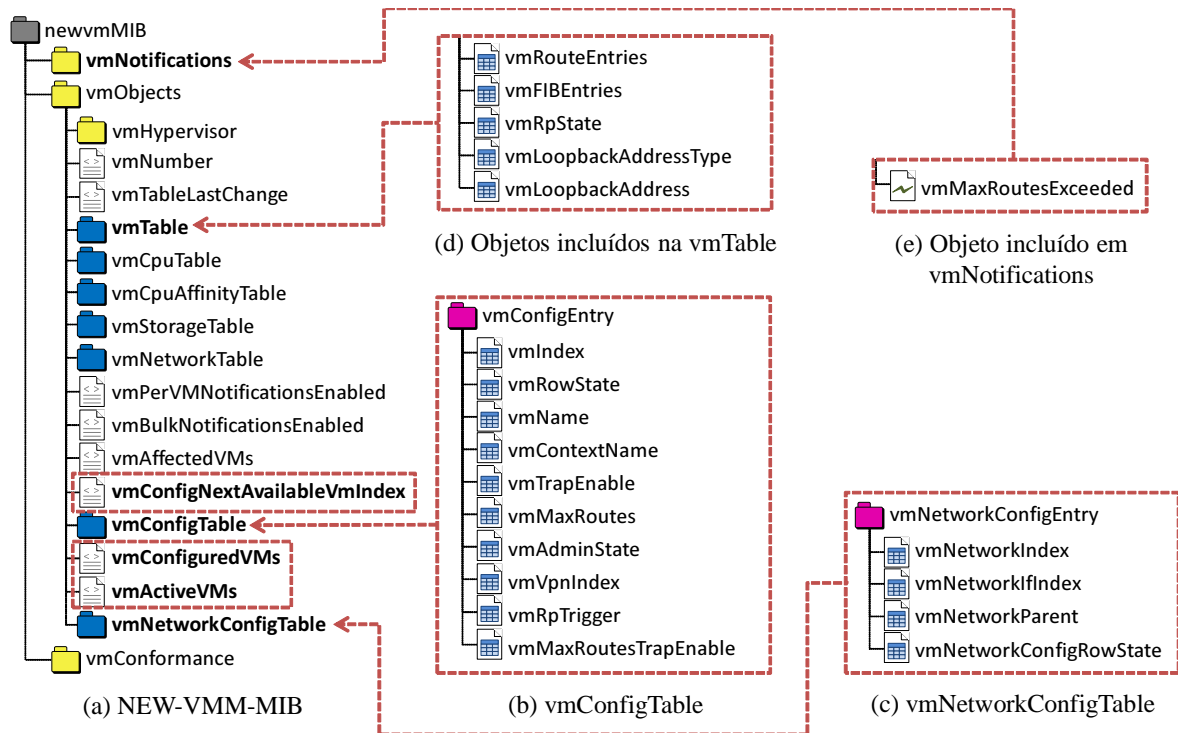
Como ponto de partida para definir um modelo de dados, foram analisados a estrutura, os objetos e as funcionalidades da VR-MIB. Conforme abordado anteriormente, a VR-MIB está desatualizada, o que dificulta e desencoraja a sua utilização em projetos atuais. Por outro lado, a VMM-MIB é uma proposta em andamento que foca no gerenciamento de máquinas virtuais controladas por um *hypervisor*. No entanto, a VMM-MIB atual não tem suporte à configuração de VRs.

Para resolver estas questões, uma nova VMM-MIB, denominada NEW-VMM-MIB, é proposta neste trabalho, usando-se a VMM-MIB como base e incorporando objetos da VR-MIB para permitir o gerenciamento apropriado de VRs. Para conseguir isso, foi necessário adicionar tabelas e objetos da VR-MIB, seguindo o padrão de nomenclatura da VMM-MIB, mudar permissões de acesso de alguns objetos da VMM-MIB, e migrar alguns objetos de suas tabelas originais para outras.

Conforme mostrado na Figura 4.2(a), a NEW-VMM-MIB tem a mesma estrutura básica da VMM-MIB, com a adição de duas novas tabelas (`vmConfigTable` e `vmNetworkConfigTable`) e três objetos escalares (`vmConfigNextAvailableVmIndex`, `vmConfiguredVMs` e `vmActiveVMs`) adaptados da VR-MIB. O `vmConfigNextAvailableVmIndex` foi adaptado do `vrConfigNextAvailableVrId` e tem por objetivo fornecer o próximo índice de VM a ser usado para criar uma entrada na tabela `vmConfigTable`, o `vmConfiguredVMs` foi adaptado do `vrConfiguredVRs` e fornece o número de VMs configuradas no elemento de rede, e o `vmActiveVMs` foi adaptado do `vrActiveVRs` e fornece o número de VMs ativas no elemento de rede.

Com relação às tabelas adicionadas, a `vmConfigTable` (Figura 4.2(b)), inspirada na VR-MIB, possui objetos com permissão *read-create* para agregar os recursos de configuração de VRs que faltavam na VMM-MIB, como por exemplo, criação e remoção de VR. De modo análogo, a `vmNetworkConfigTable` (Figura 4.2(c)) também foi incluída com permissão *read-create* em seus objetos, para possibilitar a configuração de interfaces de rede de VRs.

Figura 4.2 – Estrutura da NEW-VMM-MIB com novos objetos e tabelas



Fonte: do autor (2015).

Visando suportar todas as funcionalidades existentes na antiga VR-MIB, informações de roteamento foram incluídas na VMM-MIB adicionando-se 5 objetos à tabela `vmTable`, listados na Figura 4.2(d), com base na tabela `vrStatTable` da VR-MIB. São eles: `vmRouteEntries`, `vmFIBEntries`, `vmRpState`, `vmLoopbackAddressType` e `vmLoopbackAddress`. Análogo à VR-MIB, caso a quantidade de rotas de um VR exceda o limite máximo, uma notificação poderá ser gerada utilizando-se o objeto `vmMaxRoutesExceeded` incluído no grupo `vmNotifications`, conforme indicado na Figura 4.2(e).

Além disso, para adequar a VMM-MIB às novas funcionalidades agregadas, os objetos `vmName` e `vmAdminState` foram migrados da tabela `vmTable` para a `vmConfigTable` e tiveram suas permissões alteradas para *read-create*, para possibilitar a criação de novas entradas na tabela. Da mesma forma, os objetos `vmNetworkIfIndex` e `vmNetworkParent` foram migrados da tabela `vmNetworkTable` para a `vmNetworkConfigTable`, também com suas permissões alteradas para *read-create*.

Como resultado, a NEW-VMM-MIB é capaz de gerenciar tanto VRs quanto VMs e, consequentemente, foi utilizada como modelo de dados para as interfaces baseadas no SNMP (v2c e v3). Para interfaces baseadas no NETCONF, é necessário que haja um modelo de dados que possua as mesmas funcionalidades da NEW-VMM-MIB, mas que seja compatível com este

protocolo. Da mesma forma que o SNMP usa um módulo MIB como modelo de dados, o NETCONF usa um módulo YANG (*Yet Another Next Generation*).

Para conhecimento, YANG é a linguagem de modelagem de dados usada para modelar os dados de configuração e estado manipulados pelo protocolo NETCONF, suas RPCs (*Remote Procedure Calls*) e suas notificações, conforme a RFC 6020 (BJORKLUND, 2010). Já o SMIV2 (*Structure of Management Information version 2*), RFC 3410 (CASE et al., 2002), é uma padronização que define os tipos de dados fundamentais, um modelo de objeto, e as regras para a escrita e revisão dos módulos MIB para uso com o SNMP.

De acordo com a RFC 6643 (SCHOENWAELDER, 2012), é possível traduzir módulos MIB, definidos com SMIV2, em módulos YANG. Sendo assim, o modelo de dados da interface de gerenciamento baseada no NETCONF foi implementado através da tradução direta da NEW-VMM-MIB para a linguagem YANG. Essa tradução foi realizada utilizando-se o programa *smidump* do conjunto de ferramentas OpenYUMA (GITHUB, 2015). A estrutura da árvore se manteve a mesma da NEW-VMM-MIB, mas as tabelas foram traduzidas em listas e os objetos em folhas. Como exemplo, a Figura 4.3(a) mostra o código em SMIV2 da tabela `vmConfigTable` da NEW-VMM-MIB e a Figura 4.3(b) mostra a mesma tabela traduzida para a linguagem YANG.

Ao contrário do SNMP e do NETCONF, a solução RWS não tem uma linguagem padronizada para o modelo de dados. No RWS, o modelo de dados é altamente associado à implementação. Tendo em vista que o Java foi a linguagem escolhida para implementar a interface de gerenciamento baseada em RWS, utilizou-se o JAXB (*Java Architecture for XML Binding*) (JAXB, 2015) para criar o modelo de dados. Na implementação, cada linha da tabela (*e.g.*, `vmConfigTable`) é representada no formato XML (*eXtensible Markup Language*), e é armazenada como um Objeto em um *Map*. O `Map<K, V>` é um objeto do pacote *java.util* que mapeia uma chave *K* para um valor *V* (Objeto), da forma um para um (1:1), e não pode conter chaves duplicadas.

Como exemplo, na tabela `vmConfigTable` a chave *K* é o índice `vmIndex` e o valor *V* é uma instância de uma classe do tipo `JavaBeans`¹ que possui como propriedades todos os campos da `vmConfigTable` (*i.e.*, `vmIndex`, `vmRowState`, `vmName`, `vmContextName`, `vmTrapEnable`, `vmMaxRoutes`, `vmAdminState`, `vmVpnIndex`, `vmRpTrigger`, `vmMaxRoutesTrapEnable`). A Figura 4.4 mostra o Diagrama de Classes²,

¹Para ser considerada `JavaBeans` a classe deve ser serializável, possuir construtor sem argumentos e suas propriedades devem ser acessíveis somente via métodos *get* e *set*.

²Um Diagrama de Classes é definido pela linguagem de modelagem UML (*Unified Modeling Language*) e representa a estrutura e o relacionamento das classes que servem de modelo para objetos.

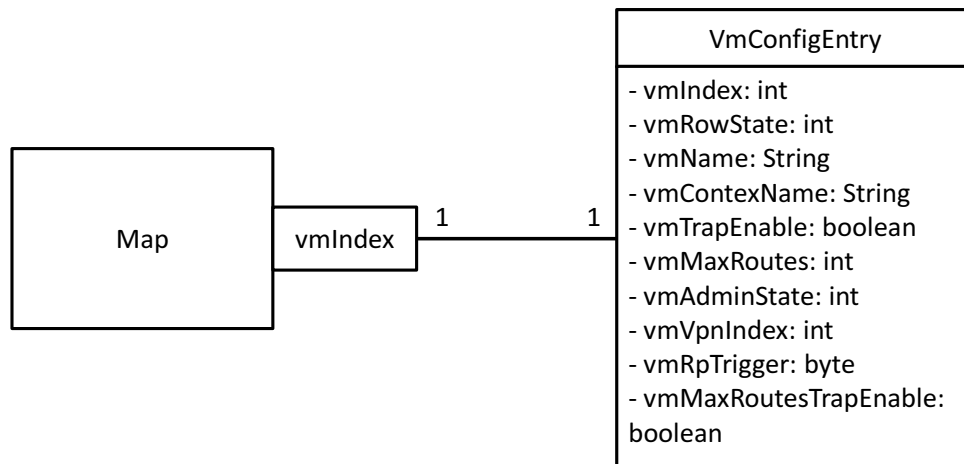
Figura 4.3 – Código da tabela vmConfigTable: SMiv2 versus YANG

(a) SMiv2	(b) YANG
<pre> 1 vmConfigEntry OBJECT-TYPE 2 SYNTAX VmConfigEntry 3 MAX-ACCESS not-accessible 4 STATUS current 5 INDEX { vmIndex } 6 ::= { vmConfigTable 1 } 7 vmIndex OBJECT-TYPE 8 SYNTAX VirtualMachineIndex 9 MAX-ACCESS not-accessible 10 STATUS current 11 ::= { vmConfigEntry 1 } 12 vmRowState OBJECT-TYPE 13 SYNTAX RowStatus 14 MAX-ACCESS read-create 15 STATUS current 16 ::= { vmConfigEntry 2 } 17 vmName OBJECT-TYPE 18 SYNTAX SnmpAdminString (SIZE (0..255)) 19 MAX-ACCESS read-create 20 STATUS current 21 ::= { vmConfigEntry 3 } 22 vmContextName OBJECT-TYPE 23 SYNTAX DisplayString 24 MAX-ACCESS read-create 25 STATUS current 26 ::= { vmConfigEntry 4 } 27 vmTrapEnable OBJECT-TYPE 28 SYNTAX TruthValue 29 MAX-ACCESS read-create 30 STATUS current 31 DEFVAL { true } 32 ::= { vmConfigEntry 5 } 33 vmMaxRoutes OBJECT-TYPE 34 SYNTAX Unsigned32 35 MAX-ACCESS read-create 36 STATUS current 37 DEFVAL { 4294967295 } 38 ::= { vmConfigEntry 6 } 39 vmAdminState OBJECT-TYPE 40 SYNTAX VirtualMachineAdminState 41 MAX-ACCESS read-create 42 STATUS current 43 ::= { vmConfigEntry 7 } 44 vmVpnIndex OBJECT-TYPE 45 SYNTAX VPNIid 46 MAX-ACCESS read-create 47 STATUS current 48 ::= { vmConfigEntry 8 } 49 vmRpTrigger OBJECT-TYPE 50 SYNTAX VmRpTriggerBitCode 51 MAX-ACCESS read-create 52 STATUS current 53 DEFVAL { { rip } } 54 ::= { vmConfigEntry 9 } 55 vmMaxRoutesTrapEnable OBJECT-TYPE 56 SYNTAX TruthValue 57 MAX-ACCESS read-create 58 STATUS current 59 DEFVAL { true } 60 ::= { vmConfigEntry 10 } </pre>	<pre> 1 list vmConfigEntry { 2 leaf vmIndex { 3 type new-vm:VirtualMachineIndex; 4 config true; 5 } 6 leaf vmRowState { 7 type smiv2:RowStatus; 8 config true; 9 } 10 leaf vmName { 11 type snmp-framework:SnmpAdminString { 12 length "0..255"; 13 } 14 config true; 15 } 16 leaf vmContextName { 17 type smiv2:DisplayString; 18 config true; 19 } 20 leaf vmTrapEnable { 21 type smiv2:TruthValue; 22 config true; 23 } 24 leaf vmMaxRoutes { 25 type uint32; 26 config true; 27 } 28 leaf vmAdminState { 29 type new-vm:VirtualMachineAdminState; 30 config true; 31 } 32 leaf vmVpnIndex { 33 type vpn-tc:VPNIid; 34 config true; 35 } 36 leaf vmRpTrigger { 37 type new-vm:VmRpTriggerBitCode; 38 config true; 39 } 40 leaf vmMaxRoutesTrapEnable { 41 type smiv2:TruthValue; 42 config true; 43 } 44 } </pre>

Fonte: do autor (2015).

referente ao exemplo anterior, que modela a utilização da classe `Map` para o armazenamento da tabela `vmConfigTable`, implementada pela classe `JavaBeans VmConfigEntry`.

Figura 4.4 – Diagrama de classes do modelo de dados para o armazenamento da tabela `vmConfigTable` em um `Map`



Fonte: do autor (2015).

Apesar das particularidades de cada modelo de dados, a informação modelada é essencialmente a mesma. As principais diferenças entre as interfaces de gerenciamento propostas são associadas aos protocolos de gerenciamento utilizados, *i.e.*, SNMP, NETCONF, e RWS. Em seguida, as mensagens trocadas entre um gerente (ou cliente) e um agente (ou servidor) para cada interface (*i.e.*, fluxo de mensagens) são discutidas.

4.3 Fluxo de Mensagens

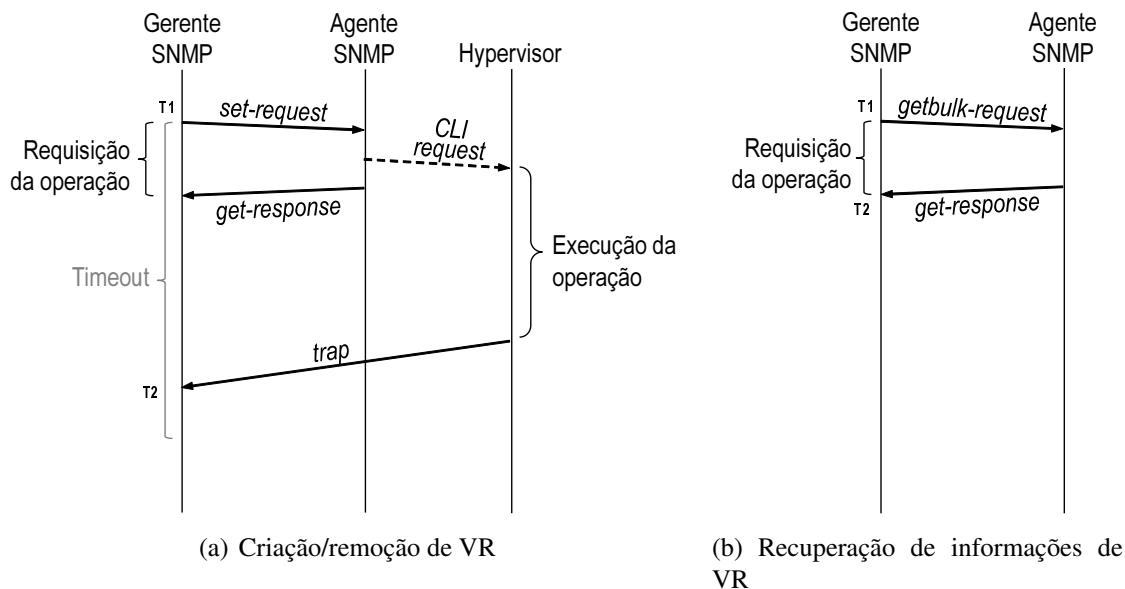
Nesta seção será apresentado o fluxo de mensagens entre gerente e agente (ou cliente e servidor) das operações de gerenciamento suportadas, *i.e.*, criação/remoção de VR e recuperação de informações de VR, para cada interface baseada nos protocolos de gerenciamento a seguir.

4.3.1 SNMPv2c

Uma troca de mensagens típicas entre um gerente e um agente SNMPv2c, usando o módulo NEW-VMM-MIB proposto, para criar/remover VR está representada na Figura 4.5(a). Inicialmente, o gerente envia uma mensagem SNMP `set-request` carregando as informações necessárias para o agente realizar a operação desejada. Como reação, o agente emite uma requisição CLI (*CLI request*) ao *hypervisor*, que cria/remove o VR. Esta requisição CLI é interna (indicada com seta tracejada), tendo em vista que o agente e o *hypervisor* estão localizados no

mesmo roteador físico. Como a ação de criação/remoção, do lado do roteador, pode durar algum tempo, um `set-request` é imediatamente respondido em paralelo com um `get-response`.

Figura 4.5 – Fluxo de mensagens da interface de gerenciamento baseada no SNMPv2c



Fonte: do autor (2015).

Quando a operação solicitada termina, uma `trap` é emitida por um gerador de `traps` em execução dentro do roteador físico, informando que o VR foi criado/removido com êxito. Tendo em vista que a mensagem `trap` pode ser perdida, pois o SNMP usa UDP (*User Datagram Protocol*) em sua comunicação, o gerente inicia um temporizador juntamente com o primeiro `set-request`. Se esse tempo expirar (*timeout*) e a confirmação da `trap` não for recebida, o gerente não tem como saber se a criação/remoção do VR foi realizada. Neste caso, são necessárias consultas adicionais ao agente para confirmar a operação.

Para recuperar informações de um VR, o gerente envia uma `getbulk-request` SNMP carregando os OID³ (*Object Identifier*) dos objetos necessários. Para recuperar as informações de um VR específico, o gerente precisa saber o `vmIndex` (índice do VR na tabela). Por outro lado, o gerente pode descobrir o `vmIndex` do VR consultando a tabela `vmConfigTable` e buscando um outro atributo, como por exemplo, o nome do VR (`vmName`). O agente responde com um `get-response` contendo os valores dos objetos solicitados, como representado na Figura 4.5(b).

³OID é uma sequência de números separados por pontos, que identifica unicamente os objetos gerenciados dentro da hierarquia da MIB.

4.3.2 SNMPv3

O SNMPv3 define uma série de capacidades relacionadas com a segurança e, dentre elas, o modelo de segurança baseado em usuários (USM – *User-based Security Model*) (BLUMENTHAL; WIJNEN, 2002) é largamente utilizado. O USM oferece serviços de autenticação e privacidade (criptografia) para o SNMP. Para essa implementação foi escolhido o HMAC-SHA-96⁴ como protocolo de autenticação (BLUMENTHAL; WIJNEN, 2002) e o CFB128-AES-128⁵ para criptografia (BLUMENTHAL; MAINO; MCCLOGHRIE, 2004).

Inicialmente, o gerente envia uma mensagem SNMP `get-request` solicitando dois parâmetros do contexto⁶: o `contextEngineID` e o `contextName` (HARRINGTON; PRESUHN; WIJNEN, 2002). Em seguida, o agente responde com uma mensagem `reportPDU` carregando, além das informações de contexto, os parâmetros relacionados com a segurança em seu cabeçalho. Os pacotes subsequentes contêm as mesmas mensagens trocadas pelo protocolo SNMPv2c, mas são criptografadas pelo AES e contêm parâmetros de segurança do USM no cabeçalho. O fluxo de mensagens referente a operação de criação/remoção de VR é mostrado na Figura 4.6(a) e o da operação de recuperação de informações de VR na Figura 4.6(b).

4.3.3 NETCONF

Para a interface de gerenciamento baseada no NETCONF, foi escolhido o SSH (*Secure Shell*) como mecanismo de transporte seguro⁷ (WASSERMAN, 2011), fornecendo autenticação, integridade de dados, confidencialidade e proteção contra ataque de repetição⁸. Por uma questão de simplicidade, o SSH não está representado no fluxo de mensagens.

Para executar uma ação de configuração (*i.e.*, criação ou remoção de VR), o cliente (gerente) NETCONF estabelece uma sessão com o servidor (agente), mas somente se não exis-

⁴O protocolo HMAC-SHA-96 utiliza o mecanismo de autenticação de mensagens HMAC (*Hash-based Message Authentication Code*) (KRAWCZYK; BELLARE; CANETTI, 1997) em conjunto com a função *hash* criptográfica SHA (*Secure Hash Algorithm*) (NIST, 2015), truncando a saída para 96 bits.

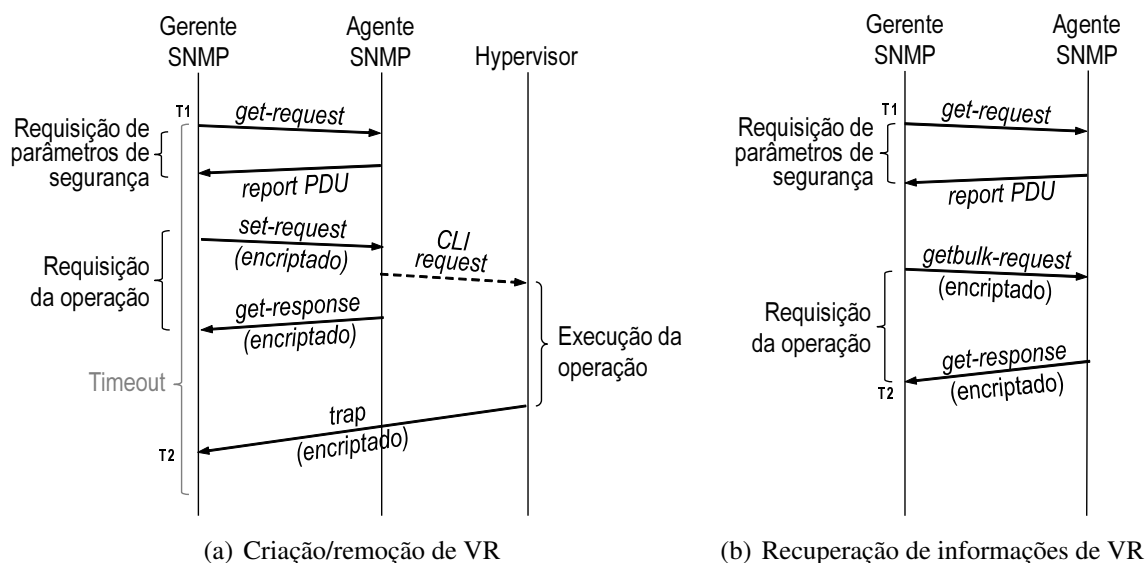
⁵O protocolo de criptografia simétrica CFB128-AES-128 utiliza o modo de cifra em blocos CFB (*Cipher Feedback*) (DWORKIN, 2001) com bloco de 128 bits, e usa o algoritmo de criptografia AES (*Advanced Encryption Standard*) (NIST, 2001) com chave criptográfica de 128 bits.

⁶Um contexto SNMP é uma coleção de informações de gerenciamento acessíveis por uma entidade SNMP. Um contexto é identificado unicamente pelo valor `contextEngineID` da entidade que hospeda as informações de gerenciamento e pelo `contextName` que, além de ser único dentro de uma entidade SNMP, é usado para nomear um contexto (HARRINGTON; PRESUHN; WIJNEN, 2002).

⁷Outras possibilidades de transporte seguro, porém menos usuais, são o NETCONF sobre TLS (*Transport Layer Security*) (BADRA, 2009), NETCONF sobre SOAP (GODDARD, 2006) e NETCONF sobre BEEP (*Blocks Extensible Exchange Protocol*) (LEAR; CROZIER, 2006).

⁸Ataque de repetição é uma forma de ataque de rede onde dados válidos são capturados e retransmitidos posteriormente de forma maliciosa e fraudulenta, visando obter acesso indevido ou causar negação de serviço (*DoS – Deny of Service*).

Figura 4.6 – Fluxo de mensagens da interface de gerenciamento baseada no SNMPv3



Fonte: do autor (2015).

tir uma sessão previamente estabelecida. O estabelecimento da sessão envolve a abertura de uma conexão TCP (omitida nos diagramas por simplicidade) e a subsequente troca de mensagens <hello>. Após o estabelecimento da sessão, o cliente NETCONF emite uma mensagem <rpc> com a operação <edit-config> contendo dados do VR (em formato XML). Depois de um <commit>, essa configuração é aplicada ao *datastore*⁹ em execução¹⁰, acionando o *hypervisor* para efetivamente criar ou remover o VR, conforme mostrado na Figura 4.7(a). A distinção entre a ação de criar ou de remover um VR é feita definindo-se o atributo XML *operation* como sendo "create" ou "delete", respectivamente. Se a operação for concluída com sucesso, o servidor responde com uma mensagem <rpc-reply> contendo um elemento <ok>.

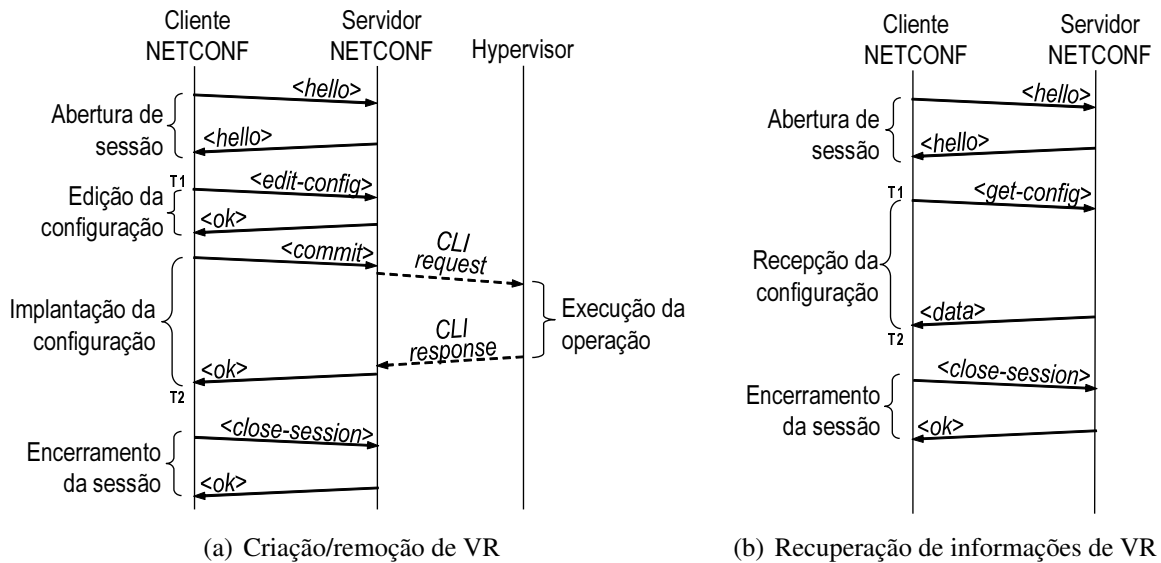
Como representado na Figura 4.7(b), para recuperar informações de um VR, o cliente tem que enviar uma mensagem <rpc> com a operação <get-config> para o servidor NETCONF. O servidor responde com uma mensagem <rpc-reply> contendo o elemento <data>, se a operação for bem sucedida, ou <rpc-error>, no caso de insucesso. Os resultados da consulta são enviados dentro do elemento <data> da mensagem de resposta.

Depois de todas as configurações/consultas, a sessão NETCONF pode ser fechada, através do envio de uma mensagem <rpc> com a operação <close-session> para o servidor NETCONF.

⁹Um *datastore* é definido como um espaço conceitual para armazenar e acessar a informação, podendo ser implementado utilizando-se arquivos, banco de dados, locais de memória, ou suas combinações (ENNS et al., 2011).

¹⁰No NETCONF, o *datastore* em execução é aquele que mantém armazenado na íntegra a configuração atualmente ativa no dispositivo (ENNS et al., 2011).

Figura 4.7 – Fluxo de mensagens da interface de gerenciamento baseada no NETCONF



Fonte: do autor (2015).

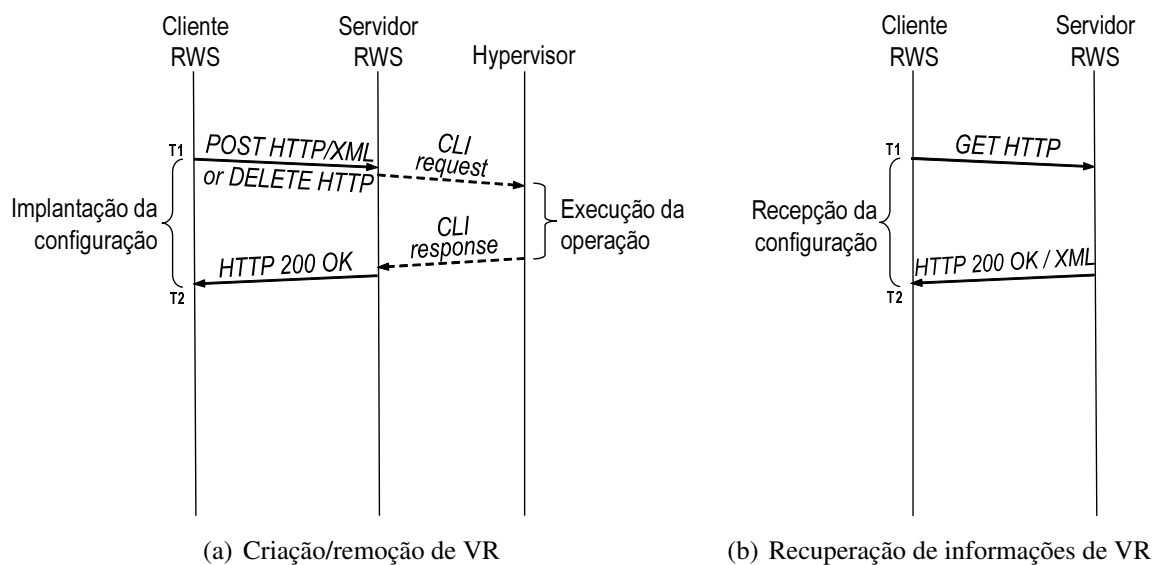
4.3.4 RESTful Web Services

A interface de gerenciamento RWS é baseada na arquitetura REST (*Representational State Transfer*) (FIELDING, 2000). Esta arquitetura utiliza métodos HTTP (e.g., GET, POST e DELETE) como verbos para solicitar um recurso identificado por um URI (*Uniform Resource Identifier*) (FIELDING; RESCHKE, 2014a; BERNERS-LEE; FIELDING; MASINTER, 2005).

Para criar um VR, o cliente RWS envia um método POST com dados no formato HTTP, XML ou JSON (*JavaScript Object Notation*). Para possibilitar uma comparação mais justa, foi escolhido o XML, pois os dados do NETCONF também são baseados em XML. O servidor RWS processa o pedido, armazena as informações do novo VR, e chama o *hypervisor* para criar o VR. Depois de criado o VR, o servidor responde enviando uma mensagem HTTP com código de *status* 200 OK (FIELDING; RESCHKE, 2014b), conforme mostrado na Figura 4.8(a). A remoção de VR (Figura 4.8(a)) é análoga à criação de VR, exceto que na remoção o cliente envia um método DELETE para um URI contendo o índice do VR (*vmIndex*) a ser removido.

A recuperação de informação de VR é similar a criação ou remoção de VR. Como mostrado na Figura 4.8(b), o cliente envia uma método GET para um URI que contém o índice do VR a ser consultado. O servidor pesquisa em seu *datastore* e, caso encontre o índice procurado, envia de volta uma resposta HTTP (200 OK) contendo os dados recuperados em formato XML. Se o índice consultado não existir no *datastore*, o servidor responde com código de *status* 204 No Content.

Figura 4.8 – Fluxo de mensagens da interface de gerenciamento baseada no RWS



Fonte: do autor (2015).

4.3.5 RESTful Web Services sobre HTTPS

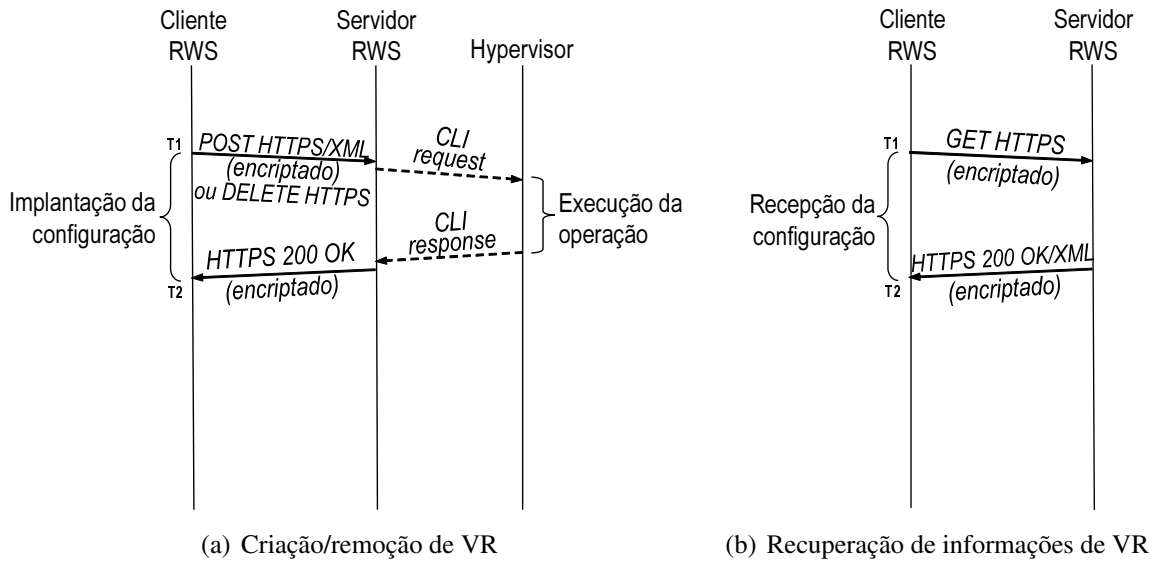
No caso da interface de gerenciamento RWS sobre HTTPS, temos que seu comportamento é análogo ao RWS, porém utiliza o protocolo seguro HTTPS no lugar do HTTP. O HTTPS (RESCORLA, 2000) é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS (*Secure Socket Layer/Transport Layer Security*), permitindo que os dados sejam transmitidos por meio de uma conexão criptografada e que a autenticidade do servidor e do cliente seja verificada por meio de certificado digital.

Na configuração do HTTPS, optou-se pelo uso do protocolo TLS (DIERKS; RESCORLA, 2008), por ser mais seguro que seu predecessor, o SSL, que possui uma série de vulnerabilidades conhecidas. Por uma questão de simplicidade, o TLS não está representado no fluxo de mensagens. A Figura 4.9(a) mostra o fluxo de mensagens para criação/remoção de VR e a Figura 4.9(b) para a recuperação de informações de VR.

4.4 Classificação

Nesta seção as interfaces de gerenciamento propostas serão classificadas em termos dos três critérios descritos na Seção 3.2: alvos, funções e abordagens de gerenciamento. Esta classificação tem por finalidade situar as interfaces propostas no universo dos projetos representativos sobre virtualização de redes, apresentados na Seção 3.1, de modo a oferecer uma visão orga-

Figura 4.9 – Fluxo de mensagens da interface de gerenciamento baseada no RWS sobre HTTPS



Fonte: do autor (2015).

nizada dos critérios de gerenciamento implementados e ajudar a identificar novos requisitos a serem considerados no futuro.

Com relação aos alvos de gerenciamento, as interfaces propostas gerenciam apenas roteadores, ou seja, nós da rede. Operações com enlaces ou redes não pertencem ao escopo destas interfaces, até o momento. A Tabela 4.1 mostra os alvos de gerenciamento suportados pelas interfaces propostas (destacada em negrito) e pelos projetos apresentados na Seção 3.1. É importante salientar que as cinco interfaces propostas foram agrupadas tendo em vista que todas têm os mesmos requisitos e, portanto, implementam os mesmos critérios de gerenciamento.

Tabela 4.1 – Classificação dos projetos e das interfaces propostas quanto aos alvos de gerenciamento

Projetos/propostas	Alvos de Gerenciamento		
	Nós	Enlaces	Redes
4WARD	✓	✓	✓
AUTOI	✓	✓	✓
FEDERICA	✓	✓	✓
ProtoGENI	✓	✓	✓
UCLP		✓	✓
VNARMS	✓	✓	✓
OpenFlow	✓		
VMWare NSX	✓	✓	✓
Interfaces propostas	✓		

Fonte: do autor (2015).

Com relação às funções de gerenciamento, tem-se que as interfaces propostas provisionam nós da rede ao executar operações de criação de VRs, ou seja, elas provisionam recursos de redes virtuais. Conforme apresentado neste capítulo, as interfaces utilizam modelos de dados baseados na NEW-VMM-MIB, e esta possui objetos de notificação (individual e em massa) para serem utilizados quando houver transição dos principais estados operacionais dos VRs provisionados. Apesar deste recurso poder ser usado para fins de monitoramento, essa funcionalidade não foi implementada nas interfaces propostas. Com relação ao interfaceamento, tem-se que esta funcionalidade é intrínseca às interfaces de gerenciamento supracitadas. A Tabela 4.2 mostra as funções de gerenciamento suportadas pelas interfaces propostas (destacada em negrito) e pelos projetos apresentados na Seção 3.1.

Tabela 4.2 – Classificação dos projetos e das interfaces propostas quanto às funções de gerenciamento

Projetos/propostas	Funções de Gerenciamento		
	Aprovisionamento	Monitoramento	Interfaceamento
4WARD	✓	✓	✓
AUTOI	✓	✓	✓
FEDERICA	✓	✓	✓
ProtoGENI	✓	✓	✓
UCLP	✓	✓	✓
VNARMS	✓		
OpenFlow	✓		
VMWare NSX	✓	✓	✓
Interfaces propostas	✓		✓

Fonte: do autor (2015).

Referente à abordagem de gerenciamento, tem-se que as interfaces de gerenciamento propostas são baseadas em protocolos que implementam o modelo gerente-agente (ou cliente-servidor). Nos experimentos optou-se, por simplicidade, pela abordagem centralizada, uma vez que foi utilizado um gerente (cliente) para executar operações de gerenciamento no roteador onde se encontrava o agente (servidor). No entanto, uma abordagem distribuída é suportada pela solução, posto que mais de um gerente (cliente) poderia comunicar-se com agentes (servidores) para executar operações de gerenciamento. Nas interfaces propostas, as operações de gerenciamento somente são executadas mediante ordem de um operador, portando elas não implementam o gerenciamento autônomo. A Tabela 4.3 mostra as abordagens de gerenciamento suportadas pelas interfaces propostas (destacada em negrito) e pelos projetos apresentados na Seção 3.1.

Tabela 4.3 – Classificação dos projetos e das interfaces propostas quanto à abordagem de gerenciamento

Projetos/propostas	Abordagens de Gerenciamento		
	Centralizado	Distribuído	Autônomo
4WARD		✓	✓
AUTOI		✓	✓
FEDERICA	✓		
ProtoGENI	✓		
UCLP		✓	
VNARMS		✓	✓
OpenFlow	✓		
VMWare NSX	✓		✓
Interfaces propostas	✓	✓	

Fonte: do autor (2015).

Neste capítulo foram apresentadas, detalhadas e classificadas cinco interfaces para o gerenciamento de roteadores físicos que hospedam roteadores virtuais. No entanto, faz-se necessária uma investigação aprofundada para determinar qual delas é a mais adequada para o fim a que se destina, considerando-se questões importantes como eficiência e consumo de recursos. No capítulo seguinte, as interfaces propostas serão avaliadas e os resultados apresentados serão discutidos.

5 AVALIAÇÃO

Neste capítulo serão avaliadas as interfaces de gerenciamento baseadas no SNMPv2c, SNMPv3, NETCONF, RWS, e RWS sobre HTTPS, com intuito de investigar qual a melhor solução para o gerenciamento de roteadores físicos que hospedam roteadores virtuais, em relação ao desempenho e à escalabilidade. Primeiramente, serão descritos o ambiente de teste e todas as ferramentas utilizadas no experimento. Em seguida, a metodologia será explicada e as métricas usadas nessa avaliação serão definidas. Por fim, serão apresentados os resultados e uma análise comparativa.

5.1 Ambiente de Teste

Os experimentos foram realizados em um computador com processador Intel Core 2 Duo @ 1,86 GHz e com 4 GB de memória RAM. O Citrix XenServer 5.6 (XENSERVEN, 2015) foi utilizado para emular um roteador físico com suporte a virtualização. O Vyatta 6.2 (BROCADE, 2015), por ser um roteador baseado em *software*, foi usado como *template* para a criação dos VRs.

O módulo NEW-VMM-MIB proposto, discutido anteriormente na Seção 4.2, foi compilado, instrumentado, e anexado a um agente Net-SNMP 5.7.2 (*snmpd*) (NET-SNMP, 2015). Ambas as interfaces SNMP (*i.e.*, v2c e v3) usam dois aplicativos Net-SNMP, o *snmpset* para enviar mensagens *SetRequest* e o *snmpgetbulk* para enviar mensagens *GetBulkRequest*. A interface SNMPv3 foi compilada para utilizar o OpenSSL (OPENSSSL, 2015), que implementa o protocolo SSL/TLS e possui bibliotecas de criptografia de uso geral. Além disso, a interface SNMPv3 foi configurada para usar SHA como protocolo de autenticação e AES para criptografia de dados.

A interface NETCONF foi implementada usando-se o conjunto de ferramentas OpenYUMA (*Open YANG Unified Modular Automation*) 2.2.5 (GITHUB, 2015). O OpenYUMA inclui um cliente (*yangcli*) e um servidor (*netconfd*) NETCONF, um compilador de módulo YANG (*yangdump*), e uma biblioteca de instrumentação de servidor (SIL - *Server Instrumentation Library*). Foi realizada a tradução da NEW-VMM-MIB de SMIV2 para a linguagem YANG, de acordo com a RFC6643 (SCHOENWAELDER, 2012), e o módulo YANG resultante foi compilado usando-se a ferramenta *yangdump*. Em seguida, o código SIL gerado foi instrumentado e anexado a um agente NETCONF.

Ambas as interfaces baseadas no RWS (sobre HTTP e sobre HTTPS) foram implantadas usando-se o Jersey (JERSEY, 2015), uma implementação do JAX-RS (JAVA.NET, 2015), que

é uma API Java para *RESTful Web Services*. Utilizou-se o Apache Tomcat 8.0.11 (APACHE, 2015) com o Java SE Runtime Environment 1.8.0_20 (ORACLE, 2015a) como servidor para a execução do *servlet* Jersey, e o cURL 7.37.1 (CURL, 2015) como cliente para enviar mensagens a uma URL (*Uniform Resource Locator*) usando os métodos HTTP `POST`, `GET`, ou `DELETE`.

Para o RWS sobre HTTPS, foram gerados um par de chaves (pública e privada) e um certificado digital X.509 v3 (ITU-T, 2012) auto-assinado. O suporte ao SSL foi habilitado no Tomcat utilizando-se a configuração padrão do conector¹ HTTP para a porta 8443, adicionando-se as informações da chave gerada. Cabe ressaltar que a configuração padrão do conector HTTP utiliza o JSSE (*Java Secure Socket Extensions*) (ORACLE, 2015b) do Java para suporte ao protocolo SSL/TLS e, portanto, não utiliza o pacote OpenSSL como no caso do SNMPv3. Para o envio das mensagens HTTPS foi utilizado o cURL com o certificado gerado incluído via linha de comando.

Como característica comum, todas as implementações foram integradas à plataforma de virtualização utilizando o comando `xe`, que é a CLI de gerenciamento nativa do XenServer. Como exemplo, para a criação de um roteador virtual de nome VR e baseado no *template* Vyatta pode ser utilizado o comando `xe vm-install new-name-label=VR template=vyatta-xenserver_VC6.2_i386`. Para a remoção do roteador virtual criado com nome de VR pode ser utilizado o comando `xe vm-uninstall vm=VR force=true`.

5.2 Metodologia e Métricas

Os experimentos foram realizados de acordo com a seguinte sequência de operações: criar VR, iniciar VR (*start VR*), recuperar informações sobre o VR criado, e, finalmente, remover VR. Cada etapa, exceto '*start VR*', equivale a uma operação de gerenciamento e foi medida em relação às seguintes métricas: tempo de resposta, tempo de CPU, consumo de memória e uso da rede.

O tempo de resposta é a métrica que registra o tempo total que cada operação de gerenciamento (*i.e.*, criação, recuperação e remoção de VR) leva para ser concluída. Esta métrica tem por finalidade verificar o quão rápida cada interface executa determinada operação de gerenciamento e, assim, descobrir se existem diferenças significativas entre as soluções. O tempo de resposta é calculado considerando-se o intervalo entre o primeiro *timestamp* (T_1) e o último

¹Conector é o elemento de ligação do Tomcat ao mundo externo, permitindo que o Tomcat receba requisições, passe-as para a aplicação Web correta e retorne os resultados. Cada conector representa uma porta que o Tomcat irá escutar, aguardando as requisições

(T2), conforme indicado no fluxo de mensagens das Figuras 4.5, 4.6, 4.7, 4.8 e 4.9.

Para efetuar a medição dos tempos T_1 e T_2 foram incluídas funções de medição de tempo em locais estratégicos do código do gerente (cliente) e, em seguida, os mesmos foram recompilados. A função `gettimeofday(struct timeval *tv, NULL)` da biblioteca `<sys/time.h>` foi utilizada para retornar o tempo corrente. Além disso, para evitar atrasos induzidos pela rede, todas as operações de gerenciamento foram executadas localmente, *i.e.*, gerentes (clientes) e agentes (servidores) foram executados no mesmo computador.

O tempo de CPU e o consumo de memória são métricas que registram, respectivamente, o tempo de processamento e a quantidade de memória usados pelo agente (servidor) quando executa uma operação de gerenciamento, *e.g.*, criação de VR. Estas métricas tem por finalidade verificar o quanto de recursos (CPU e memória) é consumido por cada interface ao executar uma determinada operação de gerenciamento e, assim, avaliar a quantidade de recursos de um roteador consumida pelas soluções apresentadas. Ambas as métricas foram obtidas utilizando-se o programa `ps` do pacote *procps* (PROCPS, 2015), cujo código-fonte foi alterado e recompilado para aumentar sua precisão de 1 para 0,001. O `ps` foi utilizado para coletar o tempo de CPU e o consumo de memória dos processos dos servidores/agentes, *i.e.*, *snmpd* (SNMP), *netconfd* (NETCONF) e *java* (RWS). Os dados foram coletadas no início (T_1) e no fim (T_2) de cada operação de gerenciamento, conforme indicado no fluxo de mensagens das Figuras 4.5, 4.6, 4.7, 4.8 e 4.9.

Para efetuar a medição do tempo de CPU e do consumo de memória em T_1 e T_2 , foram incluídas chamadas de sistema para o programa `ps` com os seguintes parâmetros: ‘-C’ seguido do nome do processo do agente (servidor) a ser medido e, ‘-o cputime’ para coleta do tempo de CPU ou ‘-o pmem’ para coleta do consumo de memória RAM em porcentagem. Para ambas as métricas, foi utilizada a função `system(const char *command)` da biblioteca `<stdlib.h>` para executar o comando `ps` com os parâmetros desejados. Essas funções foram incluídas na mesma posição do código onde foram incluídas as funções de medição de tempo da métrica anterior.

O tempo de CPU foi calculado pela diferença entre o tempo de CPU final e o inicial, pois retrata o tempo de processamento durante a operação. O consumo de memória foi determinado pela última medição da memória retornada pelo `ps`, pois reflete a porcentagem da memória total do *host*. No XenServer, a memória do *host* é limitada pela memória do *Domain-0*² (*dom0*). Sendo assim, o consumo de memória foi calculado multiplicando-se a porcentagem de memória

²*Domain-0*, *dom0*, ou *Domain Zero* é o domínio inicial executado pelo *hypervisor* XenServer na inicialização e contém o sistema operacional do *host*.

usada (em decimal) pela memória total do *dom0*.

Como carga de trabalho, após 30 ciclos de medições, o número de VRs ativos³ pré-existentes é incrementado. Um ciclo é composto pela sequência de operações de gerenciamento em um único VR, *i.e.*, criação, inicialização, recuperação e remoção. Visando obter resultados estatísticos, foi utilizado um nível de confiança de 95%.

A última métrica, uso da rede, reflete o quanto de tráfego é gerado na rede pelas operações de gerenciamento das interfaces propostas. A avaliação do uso da rede tem por finalidade verificar o impacto de cada interface de gerenciamento na rede, ao realizar suas operações. Para efetuar a medição do uso da rede, foi utilizado o programa *tcpdump* (TCPDUMP, 2015) para capturar todos os pacotes de rede trocados entre cliente e servidor (*i.e.*, gerente e agente), incluindo o *overhead*⁴ do protocolo e o estabelecimento/encerramento da sessão. Após a captura, foi calculado o volume de tráfego (em *bytes*), através da soma dos tamanhos dos pacotes trafegados em cada tipo de operação de gerenciamento, *i.e.*, criação, recuperação de informações e remoção de VRs. Assim como nas métricas anteriores, este experimento foi realizado localmente para evitar problemas adversos relacionados com a rede, tal como perda de pacotes.

Diferentemente dos experimentos anteriores, para o uso da rede a carga de trabalho consiste em incrementar o número de VRs a serem criados, recuperados, ou removidos após 30 ciclos de medições, ao invés de aumentar o número de VRs ativos pré-existentes. Os resultados do uso da rede também são apresentados com um nível de confiança de 95%.

5.3 Resultados

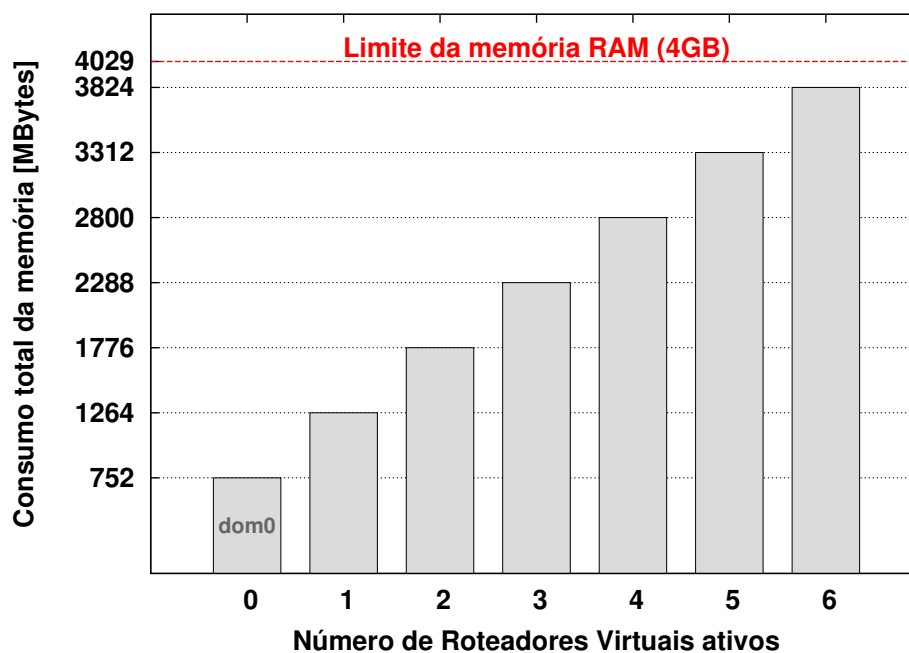
Como primeira observação, percebeu-se que o XenServer não permitia mais do que 6 VRs ativos simultaneamente. Isso acontece pois o XenServer utilizado nos experimentos aloca 752MB de RAM para o *dom0* (por padrão) e cada VR ativo, baseado no Vyatta, aloca 512MB; sendo que o total de memória RAM do ambiente de teste é 4GB. Usando-se o utilitário *xentop* do XenServer, obteve-se informações detalhadas sobre a memória alocada para o *dom0* e para os VRs ativos. A Figura 5.1 mostra o consumo total de memória RAM, neste ambiente, quando VRs são ativados.

As interfaces de gerenciamento foram avaliadas com relação a quatro métricas: tempo de resposta, tempo de CPU, consumo de memória e uso da rede. Para cada uma delas, serão apresentados os gráficos das medições e os resultados dos experimentos para cada operação de

³VR ativo é aquele que foi inicializado.

⁴Em rede, o *overhead* do protocolo refere-se a toda informação que deve ser enviada junto com os dados, e é necessária para o roteamento e transporte dos mesmos até o destino.

Figura 5.1 – Consumo de memória do XenServer com VRs ativos



Fonte: do autor (2015).

gerenciamento, *i.e.*, criação de VR, recuperação de informações de VR e remoção de VR, de acordo com a metodologia descrita na Seção 5.2.

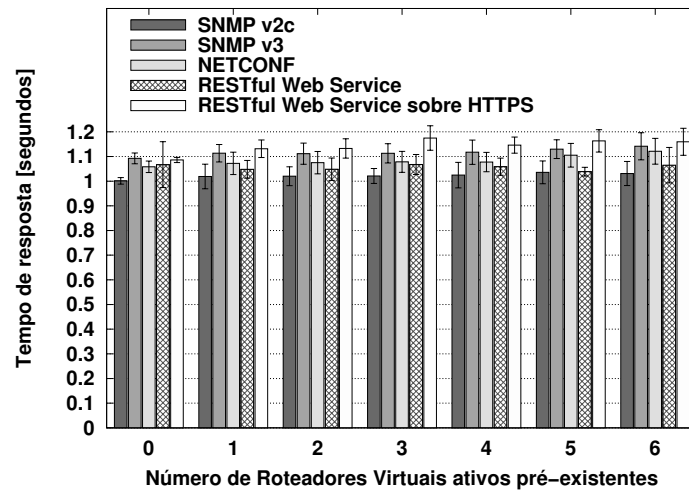
5.3.1 Tempo de resposta

A primeira métrica avaliada é o tempo de resposta. A Figura 5.2(a) representa o gráfico do tempo de resposta na criação de um VR, incrementando-se o número de VRs ativos pré-existent. As interfaces SNMPv2c e RWS apresentaram os tempos de resposta mais baixos, com resultados estatisticamente iguais. A interface RWS sobre HTTPS apresentou o maior tempo na maioria das medições, seguida pelo SNMPv3. O NETCONF apresentou tempos intermediários, situados entre o RWS e o SNMPv3.

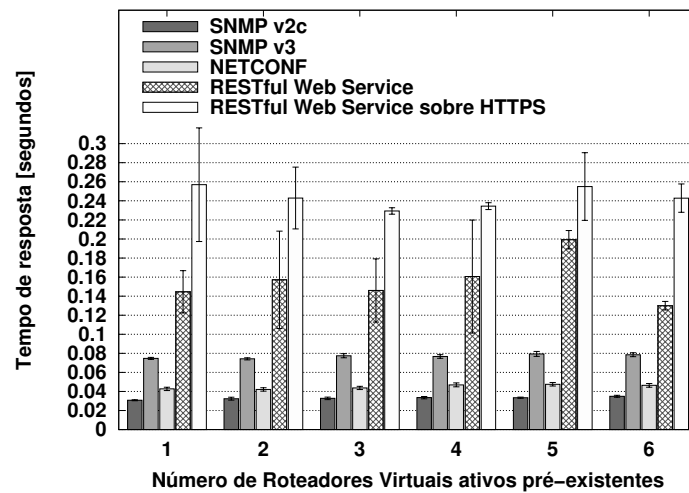
Sobre o tempo de resposta na recuperação de informações de um VR, ilustrado na Figura 5.2(b), a interface SNMPv2c obteve os menores tempos em todas as medições, seguida de perto pelo NETCONF. A interface RWS sobre HTTPS obteve os maiores tempos, seguida pelo RWS. O SNMPv3 apresentou resultados intermediários, entre o NETCONF e o RWS.

Com relação à remoção de um VR, a interface SNMPv2c apresentou menores tempos de resposta do que as demais interfaces, conforme a Figura 5.2(c). Estas apresentaram tempos bem semelhantes e, na maioria das medições, estatisticamente iguais.

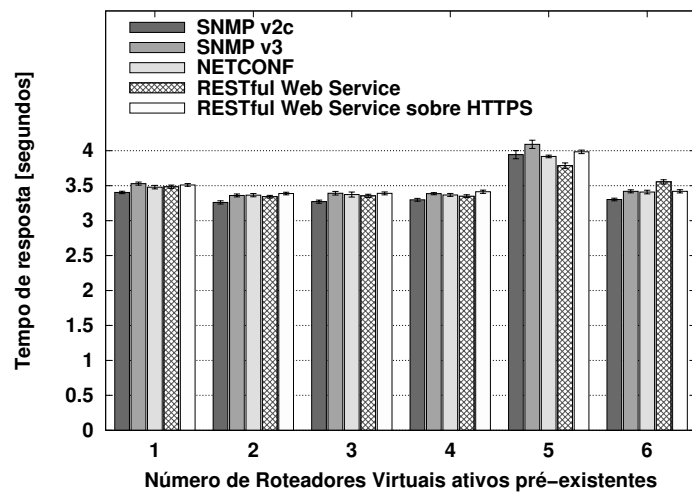
Figura 5.2 – Tempo de resposta das interfaces de gerenciamento



(a) Criação de VR



(b) Recuperação de informações de VR



(c) Remoção de VR

Fonte: do autor (2015).

5.3.2 Tempo de CPU

Com relação à métrica tempo de CPU, a Figura 5.3(a) representa os resultados das medições para criação de um VR, incrementando-se o número de VRs ativos pré-existentes. A interface SNMPv2c apresentou o menor tempo de CPU na maioria das medições, seguida pela interface SNMPv3. O NETCONF obteve tempos de CPU ligeiramente maiores que o SNMPv3 em quase todas as medições. A interface RWS obteve, em todas as medições, tempos de CPU maiores que as demais interfaces anteriormente citadas, porém obteve tempos bem inferiores ao RWS sobre HTTPS.

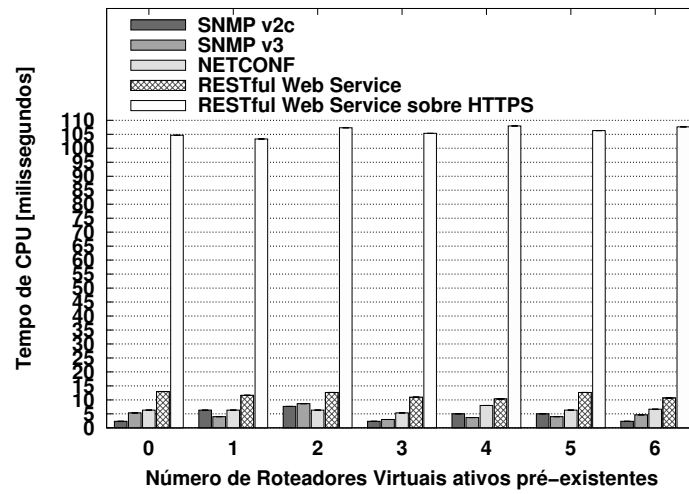
Com relação ao tempo de CPU na recuperação de informações de um VR, cujos resultados estão apresentados na Figura 5.3(b), a interface NETCONF apresentou o menor tempo nas medições com 2 ou mais VRs ativos pré-existentes. O SNMPv2c e o SNMPv3 apresentaram o segundo e o terceiro melhor tempo, sendo que em algumas medições houve a inversão desses resultados. A interface RWS obteve maior tempo de CPU que as três interfaces anteriores, ficando com o segundo pior resultado dentre as interfaces avaliadas.

Sobre o tempo de CPU na remoção de um VR, a interface SNMPv2c apresentou o menor resultado na maioria das medições, conforme observado na Figura 5.3(c). O SNMPv3 apresentou o segundo melhor resultado em todas as medições. A interface NETCONF, para 2 ou mais VRs ativos pré-existentes, obteve tempos de CPU pouco maiores que os obtidos pelo SNMPv3. O RWS, porém, apresentou tempos superiores às três interfaces anteriormente mencionadas, em todas as medições.

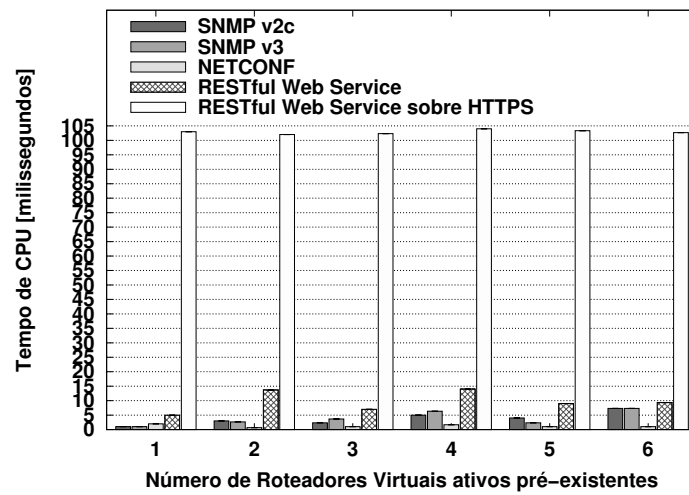
Em todos os resultados apresentados observou-se que o tempo de CPU medido varia de forma irregular à medida que a carga de trabalho é incrementada. Além disso, observou-se em todos os experimentos que a interface RWS sobre HTTPS apresentou tempos de CPU bem maiores, em torno de 110 milissegundos, que as demais interfaces, cujos tempos de CPU foram inferiores à 15 milissegundos. Esse alto processamento deve-se em grande parte à criptografia utilizada por esta interface.

Por outro lado, a interface SNMPv3, apesar de também utilizar criptografia, não apresentou tempos de CPU elevados em suas medições. Isso se deve pelo fato de que no SNMPv3 esse processamento não foi realizado pelo processo *snmpd* medido, mas sim por outro processo relacionado ao pacote OpenSSL. No caso da interface RWS sobre HTTPS, por utilizar a implementação do SSL/TLS do próprio Java, o JSSE (ORACLE, 2015b), o processamento criptográfico é todo realizado pelo processo *java* medido.

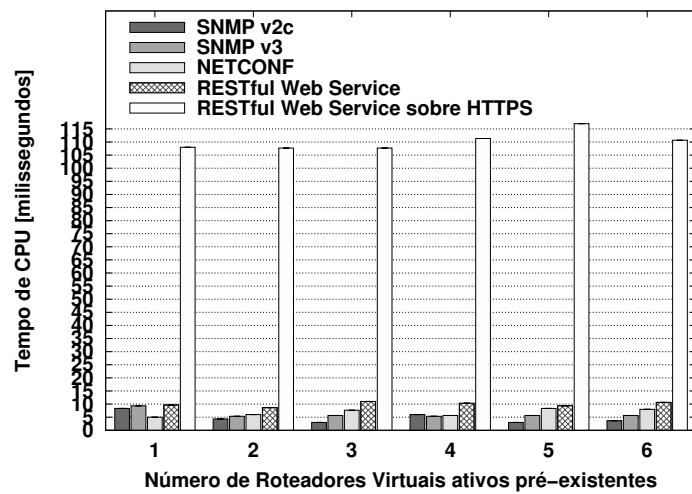
Figura 5.3 – Tempo de CPU das interfaces de gerenciamento



(a) Criação de VR



(b) Recuperação de informações de VR



(c) Remoção de VR

Fonte: do autor (2015).

5.3.3 Consumo de Memória

Com relação ao consumo de memória, os resultados mostram que as interfaces RWS e RWS sobre HTTPS consumiram muito mais memória que as demais. Quantitativamente, a interface RWS sobre HTTPS apresentou um consumo de cerca de 94MB de memória e a interface RWS em torno de 73MB, sendo que as demais interface apresentaram consumo inferiores à 5MB, conforme os gráficos da Figura 5.4. Este resultado já era esperado, pois o processo `java` é mais robusto e pesado que os processos `snmpd` e `netconfd`.

Os resultados das medições do consumo de memória das três operações de gerenciamento foram bem parecidos, conforme as Figuras 5.4(a), 5.4(b) e 5.4(c), o que mostra que o tipo de operação de gerenciamento, dentre as avaliadas, não afeta significativamente o consumo de memória.

A interface SNMPv2c apresentou o consumo mais baixo em todos os experimentos, seguido de perto pelas interfaces SNMPv3 e NETCONF. Além disso, foi observado um leve crescimento linear no consumo de memória para SNMPv2c, SNMPv3 e NETCONF, à medida que a carga de trabalho ia sendo incrementada. Diferentemente, os resultados das interfaces RWS e RWS sobre HTTPS não apresentaram um comportamento linear, mas sim pequenas oscilações.

5.3.4 Uso da Rede

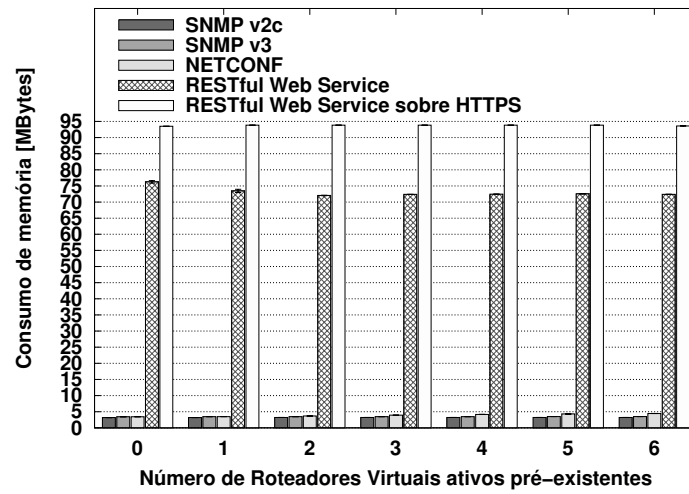
A última métrica avaliada é o uso da rede. Nos experimentos para a avaliação do uso da rede os VRs não foram ativados, tendo em vista que o processo de ativação de VR não gera tráfego na rede. Sendo assim, foi possível avaliar essa métrica para mais do que 6 VRs.

Os resultados para operações de configuração de VRs, *i.e.*, criação e remoção de VRs, mostram que para operações com menos de 4 VRs a interface baseada no NETCONF é a que gera a maior quantidade de tráfego (entre 14 e 16kBytes), conforme ilustrado nas Figuras 5.5(a) e 5.5(c). Acima de 4 VRs a interface RWS sobre HTTPS passa a ser a que mais utiliza a rede.

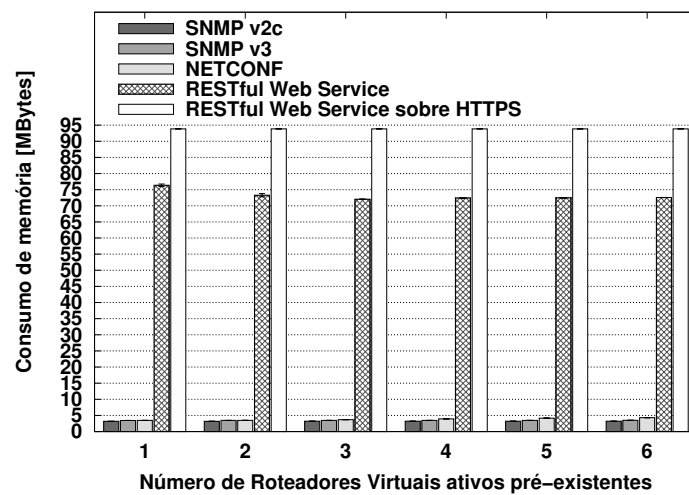
Para as operações de criação de 4 a 16 VRs, a interface NETCONF obteve o segundo maior resultado, conforme mostra a Figura 5.5(a). A partir de 16 VRs o RWS passou a gerar o segundo maior tráfego. Acima de 22 VRs, o SNMPv3 ultrapassou o NETCONF, tornando-se o terceiro maior no uso na rede.

Durante todo o experimento, *i.e.*, de 1 a 30 VRs, a interface SNMPv2c obteve o menor resultado. Porém, observando-se no gráfico que a inclinação da reta do NETCONF é menor que a do SNMPv2c, deduz-se que para uma determinada quantidade de VRs criados, a interface

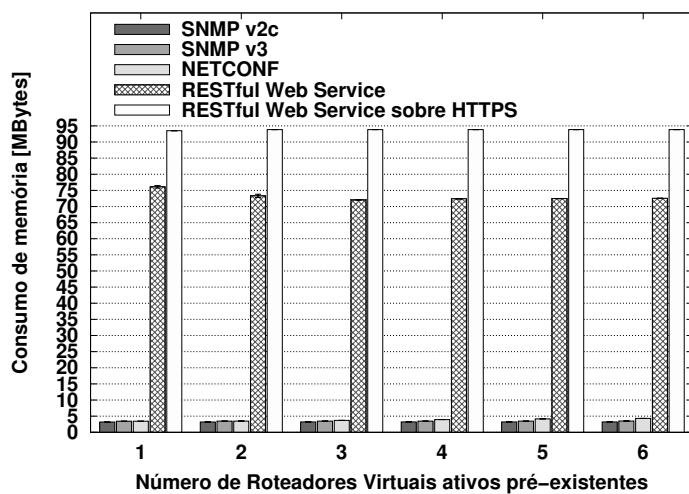
Figura 5.4 – Consumo de memória das interfaces de gerenciamento



(a) Criação de VR



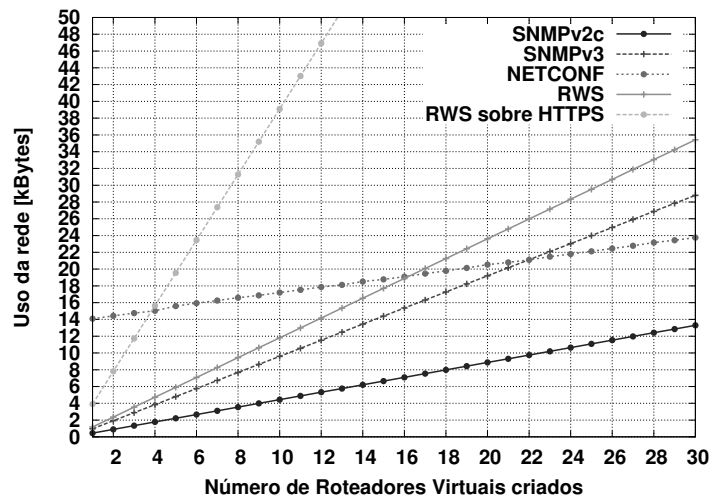
(b) Recuperação de informações de VR



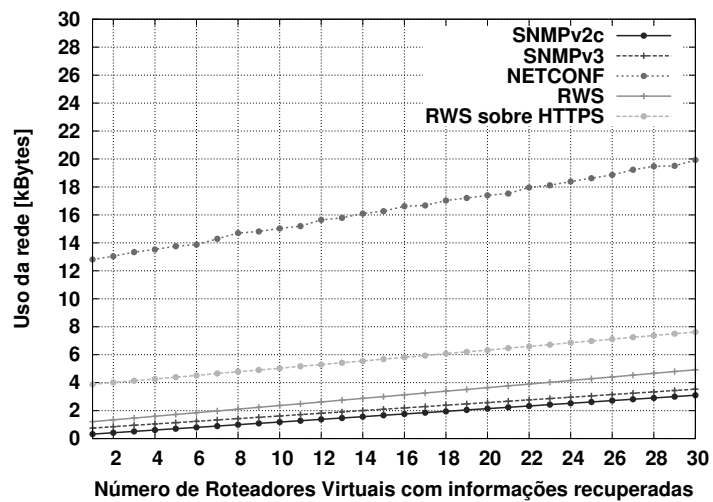
(c) Remoção de VR

Fonte: do autor (2015).

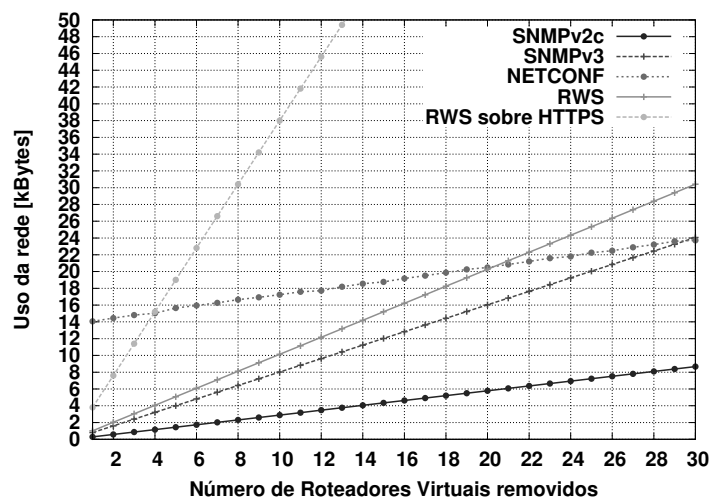
Figura 5.5 – Uso da rede pelas interfaces de gerenciamento



(a) Criação de VRs



(b) Recuperação de informações de VRs

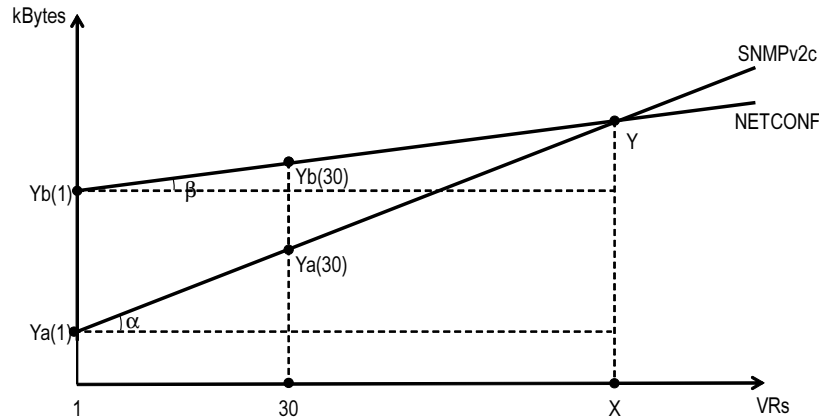


(c) Remoção de VRs

Fonte: do autor (2015).

NETCONF gerará menos tráfego do que as demais. A Figura 5.6 mostra, de forma abstrata, esta situação.

Figura 5.6 – Situação de inversão entre SNMPv2c e NETCONF



Fonte: do autor (2015).

Acima de 30 VRs, o tempo necessário para a realização do experimento torna-se muito elevado e, por isso, optou-se pela utilização de uma simples extrapolação⁵ matemática para determinar a partir de quantos VRs o NETCONF passa a ser a interface que gera o menor tráfego dentre as avaliadas. Comparando-se as tangentes dos ângulos α e β , representados na Figura 5.6, obteve-se a Equação 5.1, onde $\theta = Y_b(1) - Y_a(1)$ e $\varphi = Y_b(30) - Y_a(30)$. É importante salientar que $\theta > \varphi$ é condição para haja a interseção das retas, ou seja, o ângulo α tem que ser maior que β para haver cruzamento das retas.

$$X = 1 + \frac{29\theta}{\theta - \varphi} \quad (5.1)$$

Os parâmetros θ e φ foram calculados utilizando-se os valores dos experimentos da Figura 5.5(a). Substituindo-os na Equação 5.1 obteve-se que a interface NETCONF passa a ser a que gera o menor tráfego na rede a partir da criação de 125 VRs.

Com relação à remoção de VRs, a interface NETCONF obteve o segundo maior resultado nas operações com 4 a 20 VRs, conforme o gráfico da Figura 5.5(c). Acima de 20 VRs, o RWS passou a ser a segunda interface a gerar mais tráfego. Com 30 VRs o SNMPv3 obteve resultado semelhante ao NETCONF, o que mostra que a partir deste ponto o SNMPv3 passa a ocupar o terceiro maior lugar no grau de utilização da rede. Assim como na criação de VRs, o SNMPv2c foi a interface que gerou o menor tráfego durante todo o experimento, *i.e.*, de 1 a 30 VRs.

⁵Extrapolação é um método matemático que a partir de pontos conhecidos possibilita estimar o valor de uma função para um ponto além do intervalo de observação.

De maneira similar ao que foi feito para a Figura 5.5(a), calculando-se os parâmetros θ e φ com os valores do experimento, verificou-se que $\theta < \varphi$, indicando que não haverá cruzamento entre SNMPv2c e NETCONF ao aumentar o número de VRs removidos. Sendo assim, o SNMPv2c permanece como sendo a interface que gera o menor tráfego na remoção de VRs. E o NETCONF, para operações de remoção de mais de 30 VRs, torna-se a interface com o segundo melhor resultado.

Com relação às operações de recuperação de informações de VRs, a Figura 5.5(b) mostra que o tráfego de rede cresce uniformemente para todas as interfaces à medida que o número de VRs recuperados é incrementado. No entanto, a interface NETCONF foi a que gerou maior tráfego em comparação com as demais. O SNMPv2c apresentou o menor uso da rede em todo o experimento, seguido pelas interfaces SNMPv3, RWS e RWS sobre HTTPS, nesta ordem.

Os resultados apresentados, sobre o uso da rede, refletem o *overhead* dos protocolos utilizados nas interfaces. O SNMPv2c utiliza UDP, portanto, tem o menor *overhead*. O SNMPv3, apesar de também utilizar o UDP, foi configurado para usar autenticação e criptografia, resultando em um *overhead* maior do que a versão 2c. O RWS é baseado no HTTP, que utiliza o TCP e, portanto, tem *overhead* maior do que o SNMP. O RWS sobre HTTPS possui *overhead* ainda maior que o RWS, devido as funcionalidades de autenticação e criptografia proporcionadas pelo SSL/TLS.

O NETCONF tem o maior *overhead* de todos, pois utiliza uma codificação de dados baseada em XML para ambas as mensagens de dados e de configuração, e usa o SSH como mecanismo de transporte. Apesar disso, a interface NETCONF apresenta como vantagem um crescimento de tráfego inferior aos demais quando o número de VRs criados/removidos aumenta. Isto pode ser explicado pelo fato de que o NETCONF primeiro abre a sessão, implementa todas as configurações de VRs e, por fim, fecha a sessão. Em contraste, as interfaces baseadas no SNMP e no RWS configuram múltiplos VRs através da configuração de cada VR de forma independente. Por outro lado, para a recuperação de informações de VRs, os protocolos SNMP e RWS são mais eficientes. O SNMP utiliza mensagens `GetBulkRequest` para obter todos os objetos de uma só vez, e o RWS recupera todos os objetos armazenados no *Map* com uma única mensagem `HTTP GET`.

5.4 Comparação

Em relação ao tempo de resposta, a interface SNMPv2c apresentou os melhores resultados. Considerando a semelhança dos resultados, foi analisado o tempo de resposta médio das interfa-

ces de gerenciamento, conforme apresentado na Tabela 5.1. Na criação de VR, a interface RWS obteve resultado melhor que a interface NETCONF, seguido pelo SNMPv3 e pelo RWS sobre HTTPS. No entanto, RWS e NETCONF apresentaram tempos de resposta médios equivalentes na remoção de VR. Ao contrário da criação, na remoção, o RWS sobre HTTPS foi ligeiramente melhor que o SNMPv3.

Tabela 5.1 – Tempo de resposta médio das interface de gerenciamento

Interfaces	Criação de VR [s]	Recuperação de informações de VR [s]	Remoção de VR [s]
SNMPv2c	1.02	0.03	3.41
SNMPv3	1.12	0.08	3.53
NETCONF	1.08	0.04	3.48
RWS	1.06	0.16	3.48
RWS sobre HTTPS	1.14	0.24	3.52

Fonte: do autor (2015).

Nas operações de recuperação de informações de VR o tempo de resposta médio do NETCONF foi muito próximo ao do SNMPv2c. O pior resultado foi o da interface RWS sobre HTTPS, com o maior tempo médio, seguido pelo RWS.

Com relação ao consumo de recursos, a interface RWS sobre HTTPS apresentou o maior tempo de CPU e o maior consumo de memória, em todas as medições realizadas.

Tendo em vista os resultados do tempo de CPU terem apresentado variações não regulares à medida que o número de VRs ativos pré-existentes eram incrementados, a comparação das interfaces foi realizada considerando-se apenas o tempo máximo de CPU em cada operação de gerenciamento, conforme apresentado na Tabela 5.2.

Analisando o tempo máximo de CPU para as operações de configuração, *i.e.*, criação e remoção de VR, observou-se similaridade entre SNMPv2c e NETCONF. Nas operações de recuperação de informações de VR, o NETCONF foi a interface que consumiu menos processamento.

Referente ao consumo de memória, as interfaces SNMPv2c, SNMPv3 e NTECONF apresentaram resultados bem inferiores às interfaces baseadas no RWS, com valores inferiores a 7% dos obtidos no RWS e 5% dos obtidos no RWS sobre HTTPS. A Tabela 5.3 mostra que, em relação ao consumo médio de memória, a interface SNMPv2c foi a que menos consumiu memória, porém com resultado bem próximo do SNMPv3. O NETCONF apresentou um consumo

Tabela 5.2 – Tempo de CPU máximo das interfaces de gerenciamento

Interfaces	Criação de VR [ms]	Recuperação de informações de VR [ms]	Remoção de VR [ms]
SNMPv2c	7.7	9.3	8.3
SNMPv3	8.7	7.3	9.3
NETCONF	8.0	2.0	8.3
RWS	13.0	14.0	11.0
RWS sobre HTTPS	108.0	104.0	117.0

Fonte: do autor (2015).

médio um pouco maior que as interfaces baseadas no SNMP, ficando em terceiro lugar.

Tabela 5.3 – Consumo médio de memória das interfaces de gerenciamento

Interfaces	Criação de VR [MB]	Recuperação de informações de VR [MB]	Remoção de VR [MB]
SNMPv2c	3.2	3.2	3.2
SNMPv3	3.5	3.5	3.5
NETCONF	3.9	3.8	3.8
RWS	73.1	73.2	73.1
RWS sobre HTTPS	93.7	93.8	93.8

Fonte: do autor (2015).

Quanto ao uso da rede, a Tabela 5.4 mostra o tráfego médio gerado por cada interface ao realizar uma operação de gerenciamento (Op. Grc.) para uma determinada quantidade de VRs (#VRs). Nesta tabela não são exibidos todos os resultados presentes no gráfico da Figura 5.5, apenas os julgados importantes para fins de comparação.

Conforme explanado na Seção 5.3.4, os experimentos relacionados ao uso da rede foram realizados com até 30 VRs, no entanto, devido à característica linear dos resultados, foi realizado uma simples extrapolação das medições de modo a obter uma visão geral do uso da rede para operações com mais de 30 VRs. As Tabelas 5.5, 5.6 e 5.7 apresentam as interfaces de gerenciamento em ordem crescente de uso da rede ao aumentar a quantidade de VRs nas operações de gerenciamento.

Com relação às operações de criação de VRs (Tabela 5.5), verificou-se que a interface ba-

Tabela 5.4 – Tráfego médio gerado pelas interfaces de gerenciamento

Op. Grç.	Interfaces	Uso da rede para operações com #VRs [Kbytes]								
		1	3	4	16	17	20	21	22	30
Criação de VRs	SNMPv2c	0.4	1.3	1.8	7.1	7.5	8.9	9.3	9.8	13.3
	SNMPv3	1.0	2.9	3.8	15.4	16.3	19.2	20.2	21.1	28.8
	NETCONF	14.1	14.8	15.0	19.1	19.5	20.5	20.8	21.1	23.8
	RWS	1.2	3.5	4.7	18.9	20.1	23.6	24.8	26.0	35.4
	RWS sobre HTTPS	3.9	11.7	15.6	62.6	66.5	78.2	82.1	86.0	117.3
Recuperação de Informações de VRs	SNMPv2c	0.3	0.5	0.6	1.8	1.9	2.1	2.2	2.3	3.1
	SNMPv3	0.8	0.9	1.0	2.2	2.3	2.6	2.7	2.8	3.5
	NETCONF	12.8	13.3	13.5	16.6	16.7	17.4	17.5	18.0	19.9
	RWS	1.2	1.5	1.6	3.1	3.3	3.6	3.8	3.9	4.9
	RWS sobre HTTPS	3.9	4.1	4.3	5.8	5.9	6.3	6.5	6.6	7.6
Remoção de VRs	SNMPv2c	0.3	0.9	1.2	4.6	4.9	5.8	6.1	6.4	8.7
	SNMPv3	0.8	2.4	3.2	12.8	13.6	16.0	16.8	17.6	24.1
	NETCONF	14.1	14.8	15.0	19.2	19.5	20.5	20.8	21.2	23.7
	RWS	1.0	3.0	4.1	16.2	17.2	20.3	21.3	22.3	30.4
	RWS sobre HTTPS	3.8	11.4	15.2	60.8	64.6	76.0	79.8	83.6	114.0

Fonte: do autor (2015).

Tabela 5.5 – Ordem crescente de uso da rede das interfaces de gerenciamento para operações de criação de VRs

Ordem	$1 \leq \#VR \leq 3$	$4 \leq \#VR \leq 16$	$17 \leq \#VR \leq 21$	$22 \leq \#VR \leq 124$	$\#VR \geq 125$
1º	SNMPv2c				NETCONF
2º	SNMPv3			NETCONF	SNMPv2c
3º	RWS		NETCONF	SNMPv3	
4º	RWS sobre HTTPS	NETCONF	RWS		
5º	NETCONF	RWS sobre HTTPS			

Fonte: do autor (2015).

Tabela 5.6 – Ordem crescente de uso da rede das interfaces de gerenciamento para operações de remoção de VRs

Ordem	$1 \leq \#VR \leq 3$	$4 \leq \#VR \leq 20$	$21 \leq \#VR \leq 30$	$\#VR \geq 31$
1º	SNMPv2c			
2º	SNMPv3			NETCONF
3º	RWS		NETCONF	SNMPv3
4º	RWS sobre HTTPS	NETCONF	RWS	
5º	NETCONF	RWS sobre HTTPS		

Fonte: do autor (2015).

Tabela 5.7 – Ordem crescente de uso da rede das interfaces de gerenciamento para operações de recuperação de informações de VRs

Ordem	$\#VR \geq 1$
1º	SNMPv2c
2º	SNMPv3
3º	RWS
4º	RWS sobre HTTPS
5º	NETCONF

Fonte: do autor (2015).

seada no SNMPv2c é a que gera menos tráfego na rede, até 124 VRs criados, a partir disso o NETCONF passa a ser a interface com menor uso da rede. Inicialmente, na ordem de menor uso da rede, após o SNMPv2c vem o SNMPv3, o RWS, o RWS sobre HTTPS e o NETCONF. A partir de 4 VRs, o NETCONF passa a gerar menos tráfego do que o RWS sobre HTTPS; acima de 16 VRs, o NETCONF passa a ser melhor que o RWS; com 22 VRs, o NETCONF supera o SNMPv3; e, após 124 VRs, o NETCONF passa a ser a interface com o menor uso da rede.

Comparando-se os resultados do uso da rede para as operações de criação e remoção de VRs, Figuras 5.5(a) e 5.5(c), verificou-se que os resultados do NETCONF são muito semelhantes, contudo as demais interfaces apresentaram resultados inferiores na remoção de VRs, e essa diferença foi-se agravando à medida que o número de VRs ia sendo incrementado. Como consequência, diferente do obtido com a extrapolação dos resultados da criação de VRs, na remoção a interface SNMPv2c é sempre melhor que o NETCONF em termos de uso da rede, conforme apresentado na Tabela 5.6.

Na remoção de até 3 VRs, a sequência de interfaces que menos utilizam a rede é a mesma

da criação, *i.e.*, SNMPv2c, SNMPv3, RWS, RWS sobre HTTPS, e NETCONF. A partir de 4 VRs, o NETCONF passa a gerar menos tráfego do que o RWS sobre HTTPS; acima de 20 VRs, o NETCONF passa a ser melhor que o RWS; e, após 30 VRs, o NETCONF supera o SNMPv3, mantendo-se na segunda colocação, conforme disposto na Tabela 5.6.

Com relação ao uso da rede nas operações de recuperação de informações de VRs, a interface SNMPv2c foi a que obteve o menor volume de tráfego gerado, seguida em ordem crescente pelas interfaces SNMPv3, RWS, RWS sobre HTTPS, e NETCONF. Como as interfaces de gerenciamento apresentaram um crescimento de utilização da rede constante com o aumento do número de VRs a terem informações recuperadas (Figura 5.5(b)), a ordem classificatória apresentada na Tabela 5.7 permaneceu inalterada durante todo o experimento.

A Tabela 5.8 exibe a classificação geral das interfaces de gerenciamento em ordem crescente, orientadas de cima para baixo, para cada operação de gerenciamento avaliada, referente às métricas tempo de resposta (Tabela 5.1), tempo de CPU (Tabela 5.2), consumo de memória (Tabela 5.3) e uso da rede (Tabelas 5.5, 5.6 e 5.7).

Tabela 5.8 – Classificação das interfaces de gerenciamento de acordo com as métricas avaliadas

Métricas	Criação de VR	Recuperação de informações de VR	Remoção de VR
Tempo de resposta	SNMPv2c RWS NETCONF SNMPv3 RWS sobre HTTPS	SNMPv2c NETCONF SNMPv3 RWS RWS sobre HTTPS	SNMPv2c RWS≡NETCONF RWS sobre HTTPS SNMPv3
Tempo de CPU	SNMPv2c NETCONF SNMPv3 RWS RWS sobre HTTPS	NETCONF SNMPv3 SNMPv2c RWS RWS sobre HTTPS	SNMPv2c≡NETCONF SNMPv3 RWS RWS sobre HTTPS
Consumo de memória	SNMPv2c SNMPv3 NETCONF RWS RWS sobre HTTPS	SNMPv2c SNMPv3 NETCONF RWS RWS sobre HTTPS	SNMPv2c SNMPv3 NETCONF RWS RWS sobre HTTPS
Uso da rede (30<#VR<125)	SNMPv2c NETCONF SNMPv3 RWS RWS sobre HTTPS	SNMPv2c SNMPv3 RWS RWS sobre HTTPS NETCONF	SNMPv2c NETCONF SNMPv3 RWS RWS sobre HTTPS

Fonte: do autor (2015).

Como a ordem das interfaces apresentou variação de acordo com a quantidade de VRs operados (Tabelas 5.5 e 5.6), optou-se por exibir na Tabela 5.8 os resultados do uso da rede referentes às operações de gerenciamento com mais de 30 VRs e menos de 125 VRs, pois esse intervalo retrata um cenário de alta escalabilidade.

Como resultado da comparação, o SNMPv2c obteve o melhor resultado em quase todas as avaliações, com exceção do tempo de CPU para recuperação de informações de VRs, em que o NETCONF foi a melhor interface avaliada, seguido do SNMPv3. Na remoção de VRs, ambas as interfaces NETCONF e SNMPv2c apresentaram o melhor tempo de CPU, empatadas com o mesmo valor máximo, conforme mostrado na Tabela 5.2.

O NETCONF, por sua vez, ficou entre os dois melhores na maioria das avaliações, e quando segundo, ficou atrás somente do SNMPv2c. Nas avaliações em que ficou em terceiro na classificação, o NETCONF apresentou resultados bem similares às interfaces melhor avaliadas. Somente quanto ao uso da rede para recuperação de informações de VRs é que o NETCONF obteve um resultado pior que as demais interfaces. Isto reafirma a própria característica do NETCONF, que é um protocolo voltado para operações de configuração e, portanto, é menos otimizado para consultas.

6 CONCLUSÃO

A virtualização de redes surgiu como alternativa viável para auxiliar no desenvolvimento de novas arquiteturas de rede e para ajudar a superar a ossificação da Internet (ANDERSON et al., 2005). Desde então, esse conceito vem sendo amplamente pesquisado e difundido, tendo em vista as inúmeras possibilidades de utilização e os grandes benefícios advindos dessa tecnologia. Em um ambiente de virtualização de redes (NVE) as infraestruturas físicas (substrato) são compartilhadas entre diferentes usuários, ou prestadores de serviços, que criam múltiplas redes virtuais independentes e que coexistem de forma isolada. Além disso, em um NVE há a dissociação das funcionalidades existentes no modelo de negócio clássico, separando o papel dos tradicionais Provedores de Serviços de Internet (ISPs) em Provedores de Infraestrutura (InPs) e Prestadores de Serviços (SPs). Ao desvincular os prestadores de serviços dos provedores de infraestrutura e permitir que múltiplas arquiteturas de redes heterogêneas coabitem em um substrato físico compartilhado, a virtualização de redes proporciona flexibilidade, promove a diversidade, e promete maior segurança e capacidade de gerenciamento (CHOWDHURY; BOUTABA, 2009).

Muitos são os benefícios da virtualização de redes, entretanto sua implementação é repleta de desafios técnicos, dentre os quais o gerenciamento tem uma importância especial. O gerenciamento de NVEs é fundamental para garantir o correto funcionamento da infraestrutura física, das redes virtuais hospedadas, e dos serviços suportados pelas redes virtuais (ESTEVES; GRANVILLE; BOUTABA, 2013). No entanto, muitas questões técnicas se interpõem no caminho de sua realização bem sucedida. Um dos principais problemas de gerenciamento a ser resolvido é a definição de uma interface de gerenciamento, e protocolo, para operar roteadores físicos que hospedam roteadores virtuais. Outro problema é a dificuldade de gerenciamento de NVEs construídos sobre substratos físicos heterogêneos, *i.e.*, com equipamentos e tecnologias diferentes, pois são necessárias várias ferramentas independentes para realizar uma tarefa de gerenciamento, *e.g.*, criar um roteador virtual.

Para minimizar a dificuldade de gerenciar um conjunto diversificado de recursos físicos, interfaces de gerenciamento padronizadas devem ser definidas e avaliadas para permitir a interoperabilidade das diferentes plataformas de virtualização com as ferramentas de gerenciamento. Além disso, eficiência e escalabilidade são requisitos desejados pelos operadores de NVEs para compor suas interfaces de gerenciamento de roteadores virtuais (VRs).

6.1 Principais Contribuições e Resultados Obtidos

Uma das principais contribuições deste trabalho é a proposta de uma nova MIB, batizada de NEW-VMM-MIB, para o gerenciamento de roteadores físicos que hospedam roteadores virtuais (VRs). A NEW-VMM-MIB, inspirada na recente VMM-MIB (ASAI et al., 2014), é um modelo de dados atualizado e alinhado com os atuais esforços do IETF para o gerenciamento de ambientes virtualizados. A NEW-VMM-MIB foi desenvolvida para ser utilizada como alternativa à antiga VR-MIB (STELZER et al., 2005), tendo em vista que esta não progrediu no caminho da padronização e deixou a área de gerenciamento de VRs sem solução baseada no SNMP.

Como resultado, a NEW-VMM-MIB agrega as funcionalidades da VMM-MIB e da VR-MIB, possibilitando armazenar dados do *hypervisor*, dos recursos virtuais (*i.e.*, CPU, memória, armazenamento e interfaces de rede) e dos VRs provisionados. Além de possuir essas informações para consulta e estatística, a NEW-VMM-MIB permite operações de configuração de VRs, como por exemplo, criação e remoção de VR. Outrossim, para fins de monitoramento, essa MIB possui dois tipos de notificações (*traps*), individuais e em massa, para serem geradas quando houver transição dos principais estados operacionais dos VRs provisionados.

Como principal contribuição deste trabalho, foram implementadas, avaliadas e comparadas interfaces de gerenciamento baseadas no SNMPv2c, no SNMPv3, no NETCONF, no RESTful Web Services (RWS), e no RWS sobre HTTPS, para o gerenciamento de roteadores físicos que suportam a virtualização. Na implementação, as interfaces baseadas no SNMP utilizaram a NEW-VMM-MIB como modelo de dados. No NETCONF, o modelo de dados utilizado foi a tradução da NEW-VMM-MIB para a linguagem YANG (SCHOENWAELDER, 2012). Já nas interfaces baseadas no RWS, a estrutura da NEW-VMM-MIB foi implementada em Java, utilizando-se os recursos JAXB (JAXB, 2015) e Map. Visando analisar a eficiência, o consumo de recursos e a escalabilidade, o desempenho das interfaces foi avaliado com relação à quatro métricas: tempo de resposta, tempo de CPU, consumo de memória e uso da rede.

Comparando-se as cinco interfaces de gerenciamento investigadas, observou-se que a baseada no SNMPv2c apresentou os melhores resultados na maioria das avaliações. A segunda melhor interface, em termos de desempenho, foi a baseada no NETCONF, que alcançou resultados muito próximos à interface SNMPv2c. Embora o NETCONF tenha apresentado o maior uso da rede para operações com poucos VRs, ele demonstrou ser mais escalável do que as soluções SNMPv3, RWS e RWS sobre HTTPS. Além disso, ao contrário do SNMPv2c, o NETCONF fornece segurança, uma vez que utiliza o SSH como o mecanismo de transporte.

Sendo assim, conclui-se que a interface SNMPv2c é a mais adequada para gerenciar pequenos NVEs, que não possuem rigorosos requisitos de segurança. No entanto, o NETCONF aparece como uma alternativa promissora para compor uma interface de gerenciamento que pode ser implantada em cenários mais realistas, *i.e.*, em NVEs de grande escala, com muitos VRs, onde a segurança e a escalabilidade são preocupações altamente relevantes.

6.2 Considerações Finais e Possíveis Investigações Futuras

Apesar dos progressos alcançados, existem diversas possíveis melhorias e investigações futuras que merecem ser mencionadas. Neste trabalho, observou-se que as interfaces baseadas no RWS apresentaram alto consumo de memória em comparação às demais interfaces avaliadas, e isso é justificado pelo fato do processo `java` ser mais robusto e pesado que os das outras interfaces, *i.e.*, `snmpd` e `netconfd`. Fora isso, a interface RWS sobre HTTPS apresentou um altíssimo tempo de CPU em contraste com as demais interfaces que também implementam a autenticação/criptografia, *i.e.*, SNMPv3 e NETCONF. Isso pode ser explicado porque a configuração da interface RWS sobre HTTPS não utiliza SSL/TLS externo (OpenSSL), mas sim bibliotecas do próprio Java (JSSE – *Java Secure Socket Extensions*). Sendo assim, como oportunidade de melhoria, sugere-se uma implementação alternativa ao Java/Tomcat para as interfaces baseadas no RWS, preferencialmente uma solução baseada em C/C++ e que utilize o OpenSSL para fins de segurança (autenticação/criptografia), com objetivo de proporcionar uma avaliação mais justa.

Outra oportunidade de melhoria seria uma investigação mais aprofundada das interfaces que implementam autenticação e criptografia, *i.e.*, SNMPv3, NETCONF e RWS sobre HTTPS, tendo em vista a segurança ser um requisito cada vez mais importante em cenários realistas. Sendo assim, uma oportunidade vislumbrada seria avaliar o consumo de recursos dos processos responsáveis pela autenticação/criptografia, *e.g.*, OpenSSL e OpenSSH, de modo a melhor avaliar o impacto que esses requisitos causam no roteador físico.

Como investigação futura, sugere-se avaliar outras soluções para interface de gerenciamento, como por exemplo SOAP Web Services. Além disso, como ampliação da avaliação, sugere-se implementar outras operações de gerenciamento nas interfaces propostas e investigá-las com relação ao desempenho. No rol de novas operações de gerenciamento desejáveis estão: inicialização e desligamento de VR, criação e remoção de interface virtual de rede, recuperação de estado de VR, e cópia de VR. Outra possibilidade para o futuro é investigar a influência do *hypervisor* nos resultados dos experimentos, utilizando-se, para isso, *hypervisors* diferentes do

XenServer, como por exemplo, VMWare¹, VirtualBox², OpenVZ³, e QEMU⁴. Além disso, outras soluções de roteadores baseado em *software* podem ser experimentadas no lugar do Vyatta, como por exemplo, o Quagga⁵ e o XORP⁶.

Conforme mencionado, existem muitas possibilidades que podem ser investigadas no futuro. No entanto, este trabalho procurou apresentar uma significativa investigação dos protocolos de gerenciamento tradicionais aplicados ao gerenciamento de roteadores físicos com suporte a virtualização. Foi buscada uma melhor compreensão sobre os benefícios e limitações de tais protocolos na implementação de interfaces de gerenciamento de NVEs, atentando para as principais vantagens da padronização. Foi fornecida uma análise comparativa das interfaces propostas observando sua eficiência, consumo de recursos e escalabilidade.

Sendo assim, este trabalho procurou oferecer uma avaliação que sirva de embasamento para a escolha da interface de gerenciamento mais adequada a um determinado cenário e que incentive os fabricantes de roteadores a apoiarem uma solução padronizada. Como principal vantagem, a padronização permite que um dado produto seja compatível com as ferramentas de gerenciamento existentes, simplificando, assim, a implantação em NVEs com predominância de outros fabricantes.

¹Site oficial <<http://www.vmware.com/>>

²Site oficial <<http://www.virtualbox.org/>>

³Site oficial <<http://openvz.org/>>

⁴Site oficial <<http://www.qemu.org/>>

⁵Site oficial <<http://www.nongnu.org/quagga/>>

⁶Site oficial <<http://www.xorp.org/>>

REFERÊNCIAS

- 4WARD. *The 4WARD Project*. 2015. Disponível em: <<http://www.4ward-project.eu/index.php?s=overview>>. Acesso em: mar. 2015.
- ALKMIM, G. P.; BATISTA, D. M.; FONSECA, N. L. S. da. Mapeamento de Redes Virtuais em Substratos de Rede. In: XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC), 29., 2011, Campo Grande, MS. **Anais...** Campo Grande, MS: Universidade Federal de Mato Grosso do Sul, Faculdade de Computação, 2011. Disponível em: <http://sbrc2011.facom.ufms.br/files/anais/files/main/ST01_4.pdf>.
- ANDERSON, T. et al. Overcoming the Internet Impasse through Virtualization. **Computer**, IEEE Press, Los Alamitos, CA, USA, v. 38, n. 4, p. 34–41, abr. 2005. ISSN 0018-9162. Disponível em: <<http://dx.doi.org/10.1109/MC.2005.136>>.
- APACHE. **Tomcat**. 2015. Disponível em: <<http://tomcat.apache.org/>>. Acesso em: abr. 2015.
- ASAI, H. et al. **Management Information Base for Virtual Machines Controlled by a Hypervisor**. IETF, 2014. Internet draft. Draft-ietf-opsawg-vmm-mib-02. Disponível em: <<http://tools.ietf.org/html/draft-ietf-opsawg-vmm-mib-02>>.
- BADRA, M. **NETCONF over Transport Layer Security (TLS)**. IETF, 2009. RFC 5539 (Proposed Standard). (Request for Comments, 5539). Disponível em: <<http://www.ietf.org/rfc/rfc5539.txt>>.
- BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. **Uniform Resource Identifier (URI): Generic Syntax**. IETF, 2005. RFC 3986 (INTERNET STANDARD). (Request for Comments, 3986). Updated by RFCs 6874, 7320. Disponível em: <<http://www.ietf.org/rfc/rfc3986.txt>>.
- BJORKLUND, M. **YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)**. IETF, 2010. RFC 6020 (Proposed Standard). (Request for Comments, 6020). Disponível em: <<http://www.ietf.org/rfc/rfc6020.txt>>.
- BLUMENTHAL, U.; MAINO, F.; MCCLOGHRIE, K. **The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model**. IETF, 2004. RFC 3826 (Proposed Standard). (Request for Comments, 3826). Disponível em: <<http://www.ietf.org/rfc/rfc3826.txt>>.
- BLUMENTHAL, U.; WIJNEN, B. **User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)**. IETF, 2002. RFC 3414 (Internet Standard). (Request for Comments, 3414). Updated by RFC 5590. Disponível em: <<http://www.ietf.org/rfc/rfc3414.txt>>.
- BOUTABA, R.; GOLAB, W.; IRAQI, Y. Lightpaths on demand: A web-services-based management system. **Communications Magazine**, IEEE Press, Piscataway, NJ, USA, v. 42, n. 7, p. 101–107, jul. 2004. ISSN 0163-6804. Disponível em: <<http://dx.doi.org/10.1109/MCOM.2004.1316540>>.
- BROCADE. **SDN+NFV Develop and Enhance**. 2015. Disponível em: <<http://community.brocade.com/t5/SDN-NFV/ct-p/SdnNfv>>. Acesso em: abr. 2015.

CASE, J. et al. **Introduction and Applicability Statements for Internet-Standard Management Framework**. IETF, 2002. RFC 3410 (Informational). (Request for Comments, 3410). Disponível em: <<http://www.ietf.org/rfc/rfc3410.txt>>.

CHOWDHURY, N. M. M. K.; BOUTABA, R. Network virtualization: state of the art and research challenges. **Communications Magazine**, IEEE Press, Piscataway, NJ, USA, v. 47, n. 7, p. 20–26, jul. 2009. ISSN 0163-6804. Disponível em: <<http://dx.doi.org/10.1109/MCOM.2009.5183468>>.

_____. A survey of network virtualization. **Computer Network**, Elsevier North-Holland, Inc., New York, NY, USA, v. 54, n. 5, p. 862–876, abr. 2010. ISSN 1389-1286. Disponível em: <<http://dx.doi.org/10.1016/j.comnet.2009.10.017>>.

CLAYMAN, S.; GALIS, A.; MAMATAS, L. Monitoring virtual networks with lattice. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS (NOMS WKSPS), 12., 2010, Osaka, Japan. **Proceedings...** Osaka, Japan: IEEE/IFIP, 2010. p. 239–246. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5486569&tag=1>.

CORREIA, L. M. et al. **Architecture and Design for the Future Internet: 4WARD Project**. Springer Netherlands, 2011. ISBN 978-90-481-9345-5. Disponível em: <<http://dx.doi.org/10.1007/978-90-481-9346-2>>.

CURL. **cURL groks URLs**. 2015. Disponível em: <<http://curl.haxx.se/>>. Acesso em: abr. 2015.

DAITX, F.; ESTEVES, R.; GRANVILLE, L. On the use of SNMP as a management interface for virtual networks. In: INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM), 12., 2011, Dublin, Ireland. **Proceedings...** Dublin, Ireland: IFIP/IEEE, 2011. p. 177–184. Disponível em: <<http://dx.doi.org/10.1109/INM.2011.5990689>>.

DIERKS, T.; RESCORLA, E. **The Transport Layer Security (TLS) Protocol Version 1.2**. IETF, 2008. RFC 5246 (Proposed Standard). (Request for Comments, 5246). Updated by RFCs 5746, 5878, 6176. Disponível em: <<http://www.ietf.org/rfc/rfc5246.txt>>.

DWORKIN, M. **Recommendation for Block Cipher Modes of Operation: Methods and techniques**. NIST Special Publication 800-38A. Gaithersburg, Maryland, U.S.: National Institute of Standards and Technology, 2001. Disponível em: <<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA400014>>.

ENNS, R. et al. **Network Configuration Protocol (NETCONF)**. IETF, 2011. RFC 6241 (Proposed Standard). (Request for Comments, 6241). Disponível em: <<http://www.ietf.org/rfc/rfc6241.txt>>.

ESTEVES, R. P.; GRANVILLE, L. Z.; BOUTABA, R. On the Management of Virtual Networks. **Communications Magazine**, IEEE, IEEE Press, v. 51, n. 7, p. 2–10, jul. 2013. ISSN 0163-6804. Disponível em: <<http://dx.doi.org/10.1109/MCOM.2013.6553682>>.

FEDERICA. **Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures**. 2015. Disponível em: <<http://www.fp7-federica.eu/>>. Acesso em: mar. 2015.

FERNANDEZ, M. Evaluating OpenFlow Controller Paradigms. In: INTERNATIONAL CONFERENCE ON NETWORKS (ICN), 20., 2013, Seville, Spain. **Proceedings...** Seville, Spain: ThinkMind, 2013. p. 151–157. Disponível em: <http://www.thinkmind.org/index.php?view=article&articleid=icn_2013_7_20_10125>.

FIELDING, R.; RESCHKE, J. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**. IETF, 2014. RFC 7230 (Proposed Standard). (Request for Comments, 7230). Disponível em: <<http://www.ietf.org/rfc/rfc7230.txt>>.

_____. **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**. IETF, 2014. RFC 7231 (Proposed Standard). (Request for Comments, 7231). Disponível em: <<http://www.ietf.org/rfc/rfc7231.txt>>.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doutorado) — University of California, Irvine, 2000.

GALIS., A. et al. Management Architecture and Systems for Future Internet Networks. **Towards the Future Internet: A European Research Perspective**, IOS Press, p. 112–122, 2009. Disponível em: <<http://ebooks.iospress.nl/publication/29778>>.

GITHUB. **OpenYUMA**. 2015. Disponível em: <<https://github.com/OpenClovis/OpenYuma>>. Acesso em: abr. 2015.

GODDARD, T. **Using NETCONF over the Simple Object Access Protocol (SOAP)**. IETF, 2006. RFC 4743 (Historic). (Request for Comments, 4743). Disponível em: <<http://www.ietf.org/rfc/rfc4743.txt>>.

GRASA, E. et al. UCLPv2: a network virtualization framework built on web services [web services in telecommunications, part II]. **Communications Magazine, IEEE**, IEEE Press, v. 46, n. 3, p. 126–134, mar. 2008. ISSN 0163-6804. Disponível em: <<http://dx.doi.org/10.1109/MCOM.2008.4463783>>.

HANDLEY, M. Why the Internet only just works. **BT Technology Journal**, Kluwer Academic Publishers-Consultants Bureau, v. 24, n. 3, p. 119–129, 2006. ISSN 1358-3948. Disponível em: <<http://dx.doi.org/10.1007/s10550-006-0084-z>>.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**. IETF, 2002. RFC 3411 (INTERNET STANDARD). (Request for Comments, 3411). Updated by RFCs 5343, 5590. Disponível em: <<http://www.ietf.org/rfc/rfc3411.txt>>.

HEDSTROM, B.; WATWE, A.; SAKTHIDHARAN, S. Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions. **TLEN 5710 Capstone 2: Interdisciplinary Telecommunications Program**, University of Colorado, p. 1–13, maio 2011. Disponível em: <<http://morse.colorado.edu/~tlen5710/11s/11NETCONFvsSNMP.pdf>>.

IETF. **Layer 3 Virtual Private Networks (l3vpn)**. 2015. Disponível em: <<http://datatracker.ietf.org/wg/l3vpn/charter/>>. Acesso em: abr. 2015.

IST-AUTOI. **The Autonomic Internet**. 2015. Disponível em: <<http://www.ist-autoi.eu/>>. Acesso em: mar. 2015.

ITU-T. **Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks**. ITU, 2012. Recommendation ITU-T X.509. ISO/IEC 9594-8:2014. Disponível em: <<http://handle.itu.int/11.1002/1000/11735>>.

JAVA.NET. **Java API for RESTful Services (JAX-RS)**. 2015. Disponível em: <<https://jax-rs-spec.java.net/>>. Acesso em: abr. 2015.

JAXB. **Project JAXB**. 2015. Disponível em: <<https://jaxb.java.net/>>. Acesso em: abr. 2015.

JERSEY. **RESTful Web Services in Java**. 2015. Disponível em: <<https://jersey.java.net/>>. Acesso em: abr. 2015.

KIM, M.-S.; LEON-GARCIA, A. Autonomic Network Resource Management Using Virtual Network Concept. In: ATA, S.; HONG, C. (Ed.). **Managing Next Generation Networks and Services**. Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4773). p. 254–264. ISBN 978-3-540-75475-6. Disponível em: <http://dx.doi.org/10.1007/978-3-540-75476-3_26>.

KRAWCZYK, H.; BELLARE, M.; CANETTI, R. **HMAC: Keyed-Hashing for Message Authentication**. IETF, 1997. RFC 2104 (Informational). (Request for Comments, 2104). Updated by RFC 6151. Disponível em: <<http://www.ietf.org/rfc/rfc2104.txt>>.

KRZYWANIA, R.; DOLATA, L.; SZEGEDI, P. FEDERICA: Federated e-Infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures. **Computational Methods in Science and Technology (CMST)**, Special Issue (1), p. 19–33, 2010. Disponível em: <<http://dx.doi.org/10.12921/cmst.2010.SI.01.19-33>>.

LEAR, E.; CROZIER, K. **Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)**. IETF, 2006. RFC 4744 (Historic). (Request for Comments, 4744). Disponível em: <<http://www.ietf.org/rfc/rfc4744.txt>>.

LI, D.; HONG, X.; WITT, D. ProtoGENI, a Prototype GENI Under Security Vulnerabilities: An Experiment-Based Security Study. **Systems Journal, IEEE**, IEEE Press, v. 7, n. 3, p. 478–488, set. 2013. ISSN 1932-8184. Disponível em: <<http://dx.doi.org/10.1109/JSYST.2012.2221959>>.

MCCLOGHRIE, K.; KASTENHOLZ, F. **The Interfaces Group MIB**. IETF, 2000. RFC 2863 (Draft Standard). (Request for Comments, 2863). Disponível em: <<http://www.ietf.org/rfc/rfc2863.txt>>.

MCKEOWN, N. et al. OpenFlow: Enabling Innovation in Campus Networks. **ACM SIGCOMM Computer Communication Review**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/1355734.1355746>>.

MUNTANER, G. R. d. T. **Evaluation of OpenFlow Controllers**. 90 p. Dissertação (Mestrado) — KTH, School of Information and Communication Technology (ICT), Kista, Sweden, out. 2012. Disponível em: <<http://kth.diva-portal.org/smash/record.jsf?pid=diva2:563469>>.

NAGUIB, H. **VMware NSX Network Virtualization**. 2013. Disponível em: <<http://blogs.vmware.com/tribalknowledge/2013/03/vmware-nsx-network-virtualization.html>>. Acesso em: abr. 2015.

NET-SNMP. **Simple Network Management Protocol (SNMP)**. 2015. Disponível em: <<http://www.net-snmp.org/>>. Acesso em: abr. 2015.

NIST. Advanced Encryption Standard (AES). **Federal Information Processing Standards Publication (FIPS Pub)**, National Institute of Standards and Technology (NIST), v. 197, p. 441–0311, nov. 2001. Disponível em: <<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>>.

_____. Secure Hash Standard (SHS). **Federal Information Processing Standards Publication (FIPS Pub)**, National Institute of Standards and Technology (NIST), v. 180-4, ago. 2015. Disponível em: <<http://dx.doi.org/10.6028/NIST.FIPS.180-4>>.

OPEN NETWORKING FOUNDATION. Software-Defined Networking: The New Norm for Networks. **ONF White Paper**, Open Networking Foundation (ONF), abr. 2012. Disponível em: <<https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>>.

OPENSSL. **About the OpenSSL Project**. 2015. Disponível em: <<https://www.openssl.org/about/>>. Acesso em: maio 2015.

ORACLE. **Java SE at a Glance**. 2015. Disponível em: <<http://www.oracle.com/technetwork/java/javase/>>. Acesso em: abr. 2015.

_____. **Java Secure Socket Extension (JSSE) Reference Guide**. 2015. Disponível em: <<https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html>>. Acesso em: jun. 2015.

PRAS, A. et al. Comparing the performance of SNMP and Web services-based management. **Network and Service Management, IEEE Transactions on**, IEEE Press, v. 1, n. 2, p. 72–82, dez. 2004. ISSN 1932-4537. Disponível em: <<http://dx.doi.org/10.1109/TNSM.2004.4798292>>.

PROCPS. **The /proc file system utilities**. 2015. Disponível em: <<http://procps.sourceforge.net/>>. Acesso em: abr. 2015.

PROTOGENI. **Welcome to ProtoGENI**. 2015. Disponível em: <<http://www.protopeni.net/>>. Acesso em: abr. 2015.

RECIO, J. et al. Evolution of the User Controlled Lightpath Provisioning System. In: INTERNATIONAL CONFERENCE ON TRANSPARENT OPTICAL NETWORKS (ICTON), 7., 2005, Barcelona, Catalonia, Spain. **Proceedings...** Barcelona, Catalonia, Spain: IEEE Press, 2005. v. 1, p. 263–266. Disponível em: <<http://dx.doi.org/10.1109/ICTON.2005.1505800>>.

RESCORLA, E. **HTTP Over TLS**. IETF, 2000. RFC 2818 (Informational). (Request for Comments, 2818). Updated by RFCs 5785, 7230. Disponível em: <<http://www.ietf.org/rfc/rfc2818.txt>>.

RUBIO-LOYOLA, J. et al. Manageability of Future Internet Virtual Networks from a Practical Viewpoint. **Towards the Future Internet: Emerging Trends from European Research**, IOS Press, p. 105–114, 2010. Disponível em: <<http://dx.doi.org/10.3233/978-1-60750-539-6-105>>.

SCHOENWAEELDER, J. **Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules**. IETF, 2012. RFC 6643 (Proposed Standard). (Request for Comments, 6643). Disponível em: <<http://www.ietf.org/rfc/rfc6643.txt>>.

SHENKER, S.; WROCLAWSKI, J. **Network Element Service Specification Template**. IETF, 1997. RFC 2216 (Informational). (Request for Comments, 2216). Disponível em: <<http://www.ietf.org/rfc/rfc2216.txt>>.

SHERWOOD, R. et al. Can the Production Network Be the Testbed? In: USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI), 9., 2010, Vancouver, BC, Canada. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2010. (OSDI'10), p. 1–6. Disponível em: <<http://dl.acm.org/citation.cfm?id=1924943.1924969>>.

SOFTWARE DEFINED CLOUD NETWORKING. **VMware NSX Capabilities and Components**. 2014. Disponível em: <<http://dcx.com/2014/06/19/vmware-nsx-capabilities-and-components/>>. Acesso em: abr. 2015.

STALLINGS, W. SNMPv3: A security enhancement for SNMP. **Communications Surveys, IEEE**, IEEE, v. 1, n. 1, p. 2–17, 1998. Disponível em: <<http://dx.doi.org/10.1109/COMST.1998.5340405>>.

STELZER, E. et al. **Virtual Router Management Information Base Using SMIv2**. 2005. Internet draft. Disponível em: <<http://tools.ietf.org/html/draft-ietf-13vpn-vr-mib-04>>.

SZEGEDI, P. et al. Enabling Future Internet Research: The FEDERICA Case. **Communications Magazine, IEEE**, IEEE, v. 49, n. 7, p. 54–61, 2011. ISSN 0163-6804. Disponível em: <<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5936155>>.

TATA, K. **Definitions of Managed Objects for Virtual Router Redundancy Protocol Version 3 (VRRPv3)**. IETF, 2012. RFC 6527 (Proposed Standard). (Request for Comments, 6527). Disponível em: <<http://www.ietf.org/rfc/rfc6527.txt>>.

TCPDUMP. **TCPDUMP & LIBPCAP**. 2015. Disponível em: <<http://www.tcpdump.org/>>. Acesso em: abr. 2015.

VMWARE. **The VMware NSX Network Virtualization Platform**. 2013. Technical White Paper. Disponível em: <<https://www.vmware.com/files/pdf/products/nsx/VMware-NSX-Network-Virtualization-Platform-WP.pdf>>.

_____. **VMware NSX: A plataforma para a virtualização da rede**. 2013. Datasheet. Disponível em: <<http://www.vmware.com/files/br/pdf/products/nsx/VMware-NSX-Datasheet.pdf>>.

_____. **VMware NSX for vSphere (NSX-V) Network Virtualization**. 2013. Design Guide. Disponível em: <<http://www.vmware.com/files/br/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf>>.

_____. **NSX**. 2015. Disponível em: <<http://www.vmware.com/br/products/nsx>>. Acesso em: abr. 2015.

WASSERMAN, M. **Using the NETCONF Protocol over Secure Shell (SSH)**. IETF, 2011. RFC 6242 (Proposed Standard). (Request for Comments, 6242). Disponível em: <<http://www.ietf.org/rfc/rfc6242.txt>>.

WU, J. et al. User-managed End-to-end Lightpath Provisioning Over CA*NET 4. In: NATIONAL FIBER OPTIC ENGINEERS CONFERENCE (NFOEC), 2003, Orlando, Florida, USA. **Proceedings...** 2003. p. 275–282. Disponível em: <<http://wwwold.i2cat.net/documents/uclp2.pdf>>.

XENSER. **Open Source Virtualization**. 2015. Disponível em: <<http://www.xenserver.org/>>. Acesso em: abr. 2015.

YAP, K. et al. Towards Software Friendly Networks. In: **ACM ASIA-PACIFIC WORKSHOP ON SYSTEMS (APSYS)**, 1., 2010, New Delhi, India. **Proceedings...** New York, NY, USA: ACM, 2010. (APSys'10), p. 49–54. ISBN 978-1-4503-0195-4. Disponível em: <<http://doi.acm.org/10.1145/1851276.1851288>>.

ANEXO A ARTIGO PUBLICADO – WGRS 2014

Este artigo propõe um conjunto de extensões à VMM-MIB para permitir o gerenciamento pleno de roteadores virtuais (VRs). Esta proposta baseia-se na utilização de roteadores virtuais baseados em *software* que podem ser emulados em máquinas virtuais. As extensões propostas consistem de objetos da VR-MIB adaptados para a VMM-MIB de forma a permitir a configuração de VRs. A nova MIB proposta, denominada de NEW-VMM-MIB, foi utilizada na implementação de uma interface, baseada no SNMPv2c, para o gerenciamento de roteadores físicos que suportam virtualização. Esta interface foi avaliada em termos de tempo de resposta e consumo de recursos (CPU/memória) para operações de criação de VRs.

- **Título:**

Proposta de Modificação da VM-MIB para o Gerenciamento de Roteadores Virtuais

- **Conferência:**

XIX *Workshop* de Gerência e Operação de Redes e Serviços (WGRS - 2014)

- **Tipo:**

Trilha principal (*full-paper*)

- **Qualis:**

B5

- **URL:**

<<http://sbrc2014.ufsc.br/anais/files/wgrs/ST2-2.pdf>>

- **Data:**

5 a 9 de maio de 2014

- **Realizado em:**

Florianópolis - SC, Brasil

Proposta de Modificação da VMM-MIB para o Gerenciamento de Roteadores Virtuais

Paulo R. P. Ferraz S.¹, Rafael P. Esteves¹, Lisandro Z. Granville¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brasil

{paulo.ferraz, rpesteves, granville}@inf.ufrgs.br

Abstract. *In network virtualization environments (NVEs), the physical infrastructure is shared among different users (or service providers), which create multiple virtual networks. Virtual Routers (VRs) are created inside physical routers supporting virtualization. The management of NVEs is currently deficient because of the lack of standard management solutions for heterogeneous substrates. Thus, the most natural choice is to use SNMP, the de facto network management protocol. The first approach to manage VRs using SNMP, the VR-MIB, did not progress in the standardization path, leaving the area with no SNMP-based solution. Recently, IETF has proposed a VMM-MIB for virtual machine management that can be extended in order to support VR management. In this paper, we propose extensions to VMM-MIB to allow full VR management. These extensions enable complete configuration of VRs. In addition, we propose a VR management interface based on the extended VMM-MIB that can be used to instantiate software-based VRs inside a virtualization platform. We evaluate the proposed management interface in terms of response time and CPU/memory consumption.*

Resumo. *Em ambientes de virtualização de redes (NVEs), a infraestrutura física é compartilhada entre diferentes usuários (ou prestadores de serviços), que criam múltiplas redes virtuais. O gerenciamento de NVEs é atualmente deficiente, devido à falta de soluções padrões para substratos heterogêneos. Assim, a escolha mais natural é a utilização do SNMP, o protocolo de gerenciamento de rede de fato. A primeira abordagem para o gerenciamento de VRs, a VR-MIB, não progrediu no caminho da padronização, deixando a área sem solução baseada em SNMP. Recentemente, o IETF propôs a VMM-MIB para gerenciamento de máquina virtual, que pode ser estendida a fim de dar suporte ao gerenciamento de VRs. Neste artigo, propomos extensões para VMM-MIB para permitir o gerenciamento de VRs. Estas extensões permitem a configuração completa de VRs. Além disso, propomos uma interface de gerenciamento de VRs baseada na VMM-MIB estendida que pode ser usada para instanciar VRs baseados em software dentro de uma plataforma de virtualização. Nós avaliamos a interface de gerenciamento proposta em termos do tempo de resposta e do consumo de CPU/memória.*

1. Introdução

A virtualização de redes surgiu como alternativa viável para auxiliar no desenvolvimento de novas arquiteturas de rede e para ajudar a superar a ossificação da Internet [Anderson et al. 2005]. Em um ambiente de virtualização de redes (em inglês, Network Virtualization Environment – NVE) as infraestruturas físicas, chamadas de substrato, são compartilhadas entre diferentes usuários (ou provedores de serviço) que criam múltiplas redes virtuais, cada uma delas utilizando protocolos, esquemas de endereçamento e políticas independentes. Ao contrário da virtualização de servidores, que normalmente está localizada nos nós de borda de um ambiente de rede, a virtualização de redes ocorre principalmente no núcleo. Dessa forma, roteadores virtuais são criados e hospedados dentro de roteadores físicos, possibilitando que várias redes independentes funcionem simultaneamente, de forma isolada, sobre uma única infraestrutura física [Chowdhury and Boutaba 2009].

Atualmente, NVEs são gerenciados por meio de interfaces de linha de comando não padronizadas (em inglês, Command Line Interface – CLI) ou através de ferramentas de gerenciamento proprietárias. Sendo assim, o gerenciamento de NVEs construídos sobre substratos físicos heterogêneos (*i.e.*, com equipamentos e tecnologias diferentes) fica prejudicado, pois várias ferramentas independentes são necessárias para realizar uma tarefa de gerenciamento, como por exemplo, criar uma rede virtual. Para minimizar a dificuldade de gerenciar um conjunto diversificado de recursos físicos, interfaces de gerenciamento padronizadas devem ser definidas e avaliadas para permitir a interoperabilidade das diferentes plataformas de virtualização com as ferramentas de gerenciamento. Além disso, redes virtuais podem ser construídas a partir de substratos localizados em domínios administrativos distintos [Chowdhury and Boutaba 2010] e precisam ser gerenciadas adequadamente. Assim, neste contexto, a escolha de um protocolo de gerenciamento capaz de operar de forma eficiente e escalável roteadores físicos que hospedam roteadores virtuais é essencial.

Interfaces de gerenciamento apropriadas devem permitir que os provedores de infraestrutura (InPs – Infrastructure Providers) e de serviços (SPs – Service Providers) possam acessar, operar, manter e administrar os nós e enlaces físicos e virtuais [Esteves et al. 2013]. Em se tratando de interfaces baseadas em SNMP, existe a necessidade de MIBs *Management Interface Base* que suportem esses requisitos. No ano de 2001, foi proposta no IETF (como *Internet Draft*), uma MIB voltada para o gerenciamento de Roteadores Virtuais (VRs – Virtual Routers), denominada VR-MIB [Stelzer et al. 2003]. Apesar de diversas atualizações, a VR-MIB não se tornou um padrão e está expirada desde 2006. No entanto, recentemente, foi proposta uma MIB para o gerenciamento de Máquinas Virtuais (VMs), denominada VMM-MIB, mas sua proposta original não prevê o suporte necessário à configuração de VRs.

O objetivo deste artigo é propor um conjunto de extensões à VMM-MIB para que esta possa suportar o gerenciamento pleno de roteadores virtuais. Nossa abordagem é baseada na utilização de roteadores virtuais baseados em software que podem ser emulados em máquinas virtuais. Sendo assim, as extensões propostas consistem de objetos da VR-MIB adaptados para a VMM-MIB que permitem a configuração de VRs. Além disso, definimos um modelo de gerenciamento para NVEs baseado no SNMP e na plataforma de gerenciamento XenServer [XenServer 2013].

O restante deste artigo está organizado da seguinte forma. Na Seção 2, são descritas as MIBs VR-MIB e VMM-MIB, seus objetos e suas funcionalidades. Na Seção 3, apresentamos a proposta de modificação da VMM-MIB, mostrando os objetos incluídos e os alterados. Na Seção 4, é feita a avaliação da principal funcionalidade agregada à VMM-MIB, a criação de VRs. Finalmente, na Seção 5, concluímos o artigo e apresentamos direções para trabalhos futuros.

2. Trabalhos Relacionados

2.1. VR-MIB

Uma das primeiras iniciativas para o gerenciamento de redes com roteadores virtuais foi a *Virtual Router Management Information Base* (VR-MIB) [Stelzer et al. 2005]. A VR-MIB foi proposta em 2001, originalmente para a gerência de VPNs (*Virtual Privated Networks*) de camada 3, mas foi descontinuada em 2006 e não chegou a ser padronizada.

A VR-MIB define objetos para armazenar estatísticas e permitir a configuração de alto nível de roteadores virtuais, como criação, remoção e o mapeamento entre os roteadores virtuais e suas interfaces de rede virtuais.

O gerenciamento de roteadores virtuais necessita de objetos para possibilitar a configuração de VRs, a coleta de estatísticas de VRs e a associação de interfaces de rede virtuais com as físicas [Daitx et al. 2011]. Na VR-MIB mais atual (L3VPN-VR-MIB versão 4), esses 3 grupos de informações foram definidos, respectivamente, como: **vrConfig**, **vrStat** e **vrIfConfig**.

O grupo **vrConfig**, mostrado na Figura 1(a), contém informações sobre indexação, criação e exclusão de VRs. Para ajudar a estação de gerenciamento na atribuição de um novo VR sem conflito, o próximo identificador de VR disponível pode ser recuperado acessando o objeto escalar **vrConfigNextAvailableVrId** do dispositivo gerenciado. Cada configuração de roteador virtual é armazenada em uma linha da tabela **vrConfigTable**, que contém um identificador único (**vrId**), o nome do VR (**vrName**) e o status da linha (**vrRowStatus**), sendo este último usado para criar e excluir os roteadores virtuais. Além disso, essa tabela contém informações sobre configurações internas de cada dispositivo virtual, como o identificador de VPN (**vrVpnId**), o número máximo de rotas virtuais suportadas (**vrMaxRoutes**) e um gatilho para ligar ou desligar os protocolos de roteamento (**vrRpTrigger**).

O grupo **vrStat**, mostrado na Figura 1(b), contém 2 objetos escalares com informações globais de dispositivos, como o número de VRs configurados em cada nó (**vrConfiguredVRs**) e o número de VRs ativos na rede (**vrActiveVRs**). Além disso, este grupo contém uma tabela chamada **vrStatTable**, que possui os status administrativo e operacional de cada VR. Essa tabela contém diversas estatísticas como: o número atual de entradas de rota (**vrStatRouteEntries**), o número de entradas na FIB (*Forwarding Information Base*) (**vrStatFIBEntries**), o tempo decorrido após o VR entrar em operação (**vrStatUpTime**), o status operacional do VR (**vrOperStatus**), o tipo e o endereço IP de uma das interfaces do VR (**vrRouterAddressType** e **vrRouterAddress**).

O grupo **vrIfConfig**, mostrado na Figura 1(c), é composto por uma tabela usada para a configuração de interfaces de VRs (**vrIfConfigTable**). Esta tabela contém todas as associações entre os roteadores virtuais e as interfaces de rede físicas, pois possui

como índices da **vrIfConfigEntry** os identificadores **vrId** e **vrIfId**. Assim como na **vrConfigTable**, interfaces de roteadores virtuais podem ser criadas, excluídas ou modificadas editando-se a variável **vrIfConfigRowStatus**. Este objeto é usado para acionar as operações de criação e remoção ou para definir o estado de operação de cada uma das interfaces virtuais de VRs. A fim de definir uma nova interface de VR, o agente SNMP primeiro verifica se o VR está disponível e, em seguida, verifica se a interface de rede escolhida está disponível.

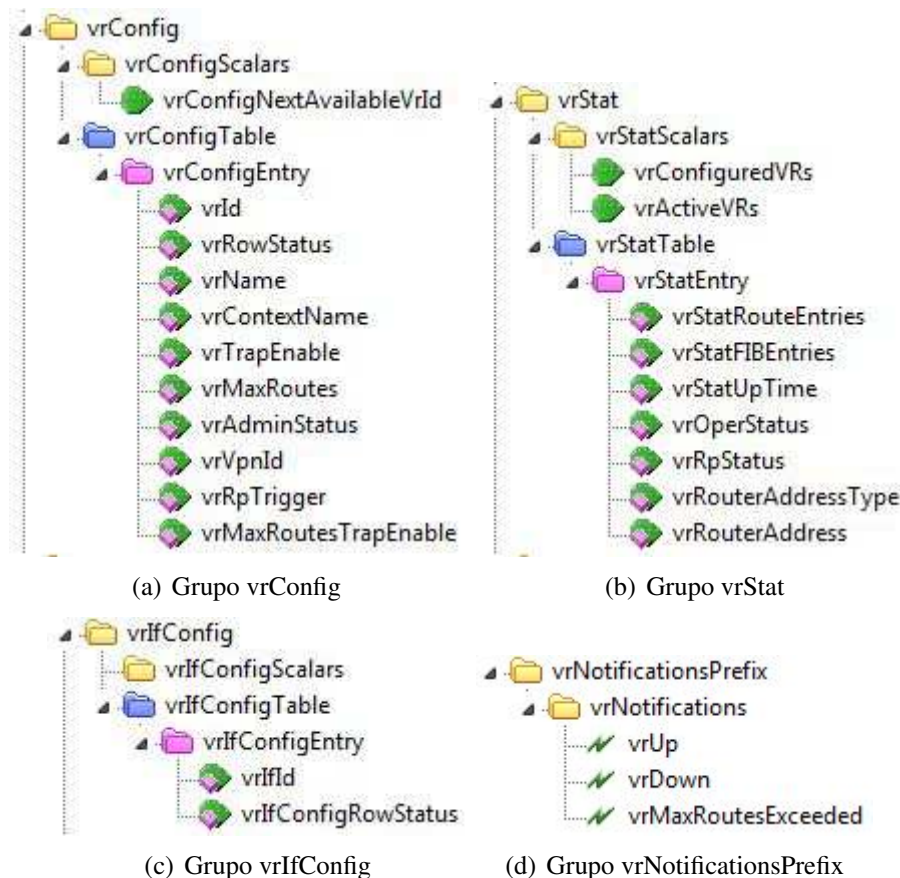


Figura 1. Grupos de informações da VR-MIB

Além dos três grupos supracitados, o grupo **vrNotificationsPrefix** (Figura 1(d)) permite que o gerente ative ou desative *traps* manipulando a variável **vrTrapEnable** da tabela **vrConfigTable**. Esse grupo é formado pelas *traps*: **vrUp**, **vrDown** e **vrMaxRoutesExceeded**.

A principal desvantagem da VR-MIB é que ela deixou de ser atualizada em 2006, fato que desencoraja sua utilização em projetos atuais. Também não há MIB que substitua suas funcionalidades no que diz respeito à configuração de roteadores virtuais. Sendo assim, a solução mais rápida para este problema é procurar uma MIB atual que possa absorver as funcionalidades da VR-MIB.

2.2. VMM-MIB

Em julho de 2012, foi proposta no IETF, como *Internet Draft*, uma MIB voltada para o gerenciamento de Máquinas Virtuais (VMs – *Virtual Machines*) controladas por um *hy-*

pervisor, com o título de *Management Information Base for Virtual Machines Controlled by a Hypervisor*, ou simplesmente VMM-MIB [Asai et al. 2013].

Um *hypervisor*, também conhecido como monitor de máquina virtual, controla várias máquinas virtuais em uma única máquina física, alocando recursos para cada VM usando tecnologias de virtualização. Portanto, este módulo MIB contém informações sobre as máquinas virtuais e seus recursos controlados por um *hypervisor*, bem como informações de *hardware* e *software* do mesmo.

O projeto desta MIB foi derivado de módulos MIB específicos de empresas, como Xen e VMWare, e de um módulo MIB que usava a interface de programação *libvirt* para acessar diferentes *hypervisors*. No entanto, esta MIB tenta generalizar os objetos gerenciados para suportar outros *hypervisors*.

A VMM-MIB possui informações relativas ao sistema e *software* de um *hypervisor*, a lista de máquinas virtuais controladas, e recursos virtuais alocados para máquinas virtuais. Sobre este último, são especificados nessa MIB quatro tipos específicos de recursos virtuais comuns a todos os *hypervisors*: CPUs (processadores), memória, interfaces de rede e dispositivos de armazenamento. A Figura 2 esquematiza os recursos físicos e virtuais dentro do *hypervisor*, e destaca o agente SNMP, que é responsável por manter a MIB informada.

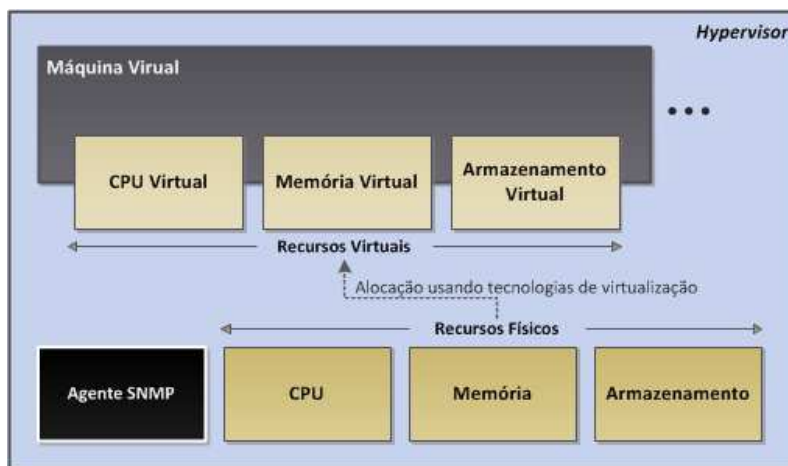


Figura 2. Alocação de recursos virtuais

A VMM-MIB está organizada em um grupo de nove escalares e cinco tabelas. Os escalares organizados sob **vmHypervisor** (Figura 3) fornecem informações básicas sobre o *hypervisor*, como a descrição do *software* (**vmHvSoftware**), a versão (**vmHvVersion**), o identificador (**vmHvObjectID**) e o tempo desde que o *hypervisor* foi reinicializado (**vmHvUpTime**). Os demais objetos escalares são: o **vmNumber**, que contém o número de VMs no *hypervisor*; o **vmTableLastChange**, que é valor do **vmHvUpTime** da última criação ou exclusão de entrada na **vmTable**; o **vmPerVMNotificationsEnable**, que habilita a geração de notificações por VMs; o **vmBulkNotificationsEnable**, que é semelhante ao anterior, mas para conjuntos de VMs; e o **vmAffectedVMs**, que contém a lista de VMs que tiveram seu estado alterado.

Em relação às tabelas, temos: a **vmTable** (Figura 4(a)), que lista as VMs que são conhecidas pelo *hypervisor*; a **vmCpuAffinityTable** (Figura 4(c)), que fornece a

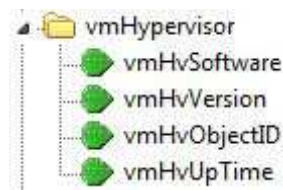


Figura 3. Escalares do grupo **vmHvSoftware** da VMM-MIB

afinidade de cada processador virtual com uma CPU física; a **vmStorageTable** (Figura 4(b)), que fornece a lista de dispositivos de armazenamento virtual e seu mapeamento com máquinas virtuais; a **vmCpuTable** (Figura 4(c)), que fornece o mapeamento entre CPUs virtuais para máquinas virtuais; e a **vmNetworkTable** (Figura 4(d)), que fornece a lista de interfaces de rede virtuais e seu mapeamento para máquinas virtuais. As tabelas **vmTable** e **vmNetworkTable**, por serem mais relevantes para este trabalho, serão descritas com mais detalhes.

A tabela **vmTable**, mostrada na Figura 4(a), contém diversas informações sobre as VMs, sendo que as principais para este trabalho são: um identificador único da VM (**vmIndex**), o nome da VM (**vmName**), o estado administrativo (**vmAdminState**), o estado operacional da VM (**vmOperState**), o número atual, mínimo e máximo de CPUs atribuídas à VM (**vmCurCpuNumber**, **vmMinCpuNumber** e **vmMaxCpuNumber**) e o tamanho atual, mínimo e máximo de memória alocada para a VM (**vmCurMem**, **vmMinMem** e **vmMaxMem**). O objeto **vmAdminState** é um parâmetro que define o modelo de estado administrativo da VM e, por isso, tem permissão *read-write*. Da mesma forma, por serem parâmetros de configuração, também possuem permissão de acesso *read-write* os objetos supracitados relacionados com CPU e memória. Os demais objetos da tabela têm acesso *read-only*.

A tabela **vmNetworkTable**, mostrada na Figura 4(d), contém informações de interfaces de redes virtuais, como um identificador único (**vmNetworkIndex**), um ponteiro para a interface corresponde na **ifTable** da IF-MIB [McCloghrie and Kastenholz 2000] (**vmNetworkIfIndex**), outro ponteiro para **ifTable** da IF-MIB no caso de haver uma interface de rede física pai correspondente (**vmNetworkParent**), o modelo de interface emulado pela interface de rede virtual (**vmNetworkModel**), e o endereço físico (*MAC Address*) (**vmNetworkPhysAddress**).

Eventos que causam transições entre os principais estados operacionais geram notificações (*traps*). A VMM-MIB define dois tipos de notificações: as notificações individuais (*per-VM*) e as notificações em massa (*bulk*). As *per-VM* levam informações mais detalhadas, contudo a escalabilidade pode ser um problema. Já o mecanismo de notificação em massa tem o objetivo de reduzir o número de notificações que são capturadas por um gerente SNMP. As notificações *per-VM*, definidas pelos objetos listados na Figura 5(a), são geradas se **vmPerVMNotificationsEnable** for verdadeiro. De modo análogo, as notificações em massa, definidas pelos objetos da Figura 5(b), são geradas se **vmBulkNotificationsEnabled** for verdadeiro.

3. Proposta de modificação da VMM-MIB

A idéia da proposta é agregar à VMM-MIB todas as funcionalidades da VR-MIB, sem que a primeira perca suas funcionalidades originais. Inicialmente, percebe-se que os escopos

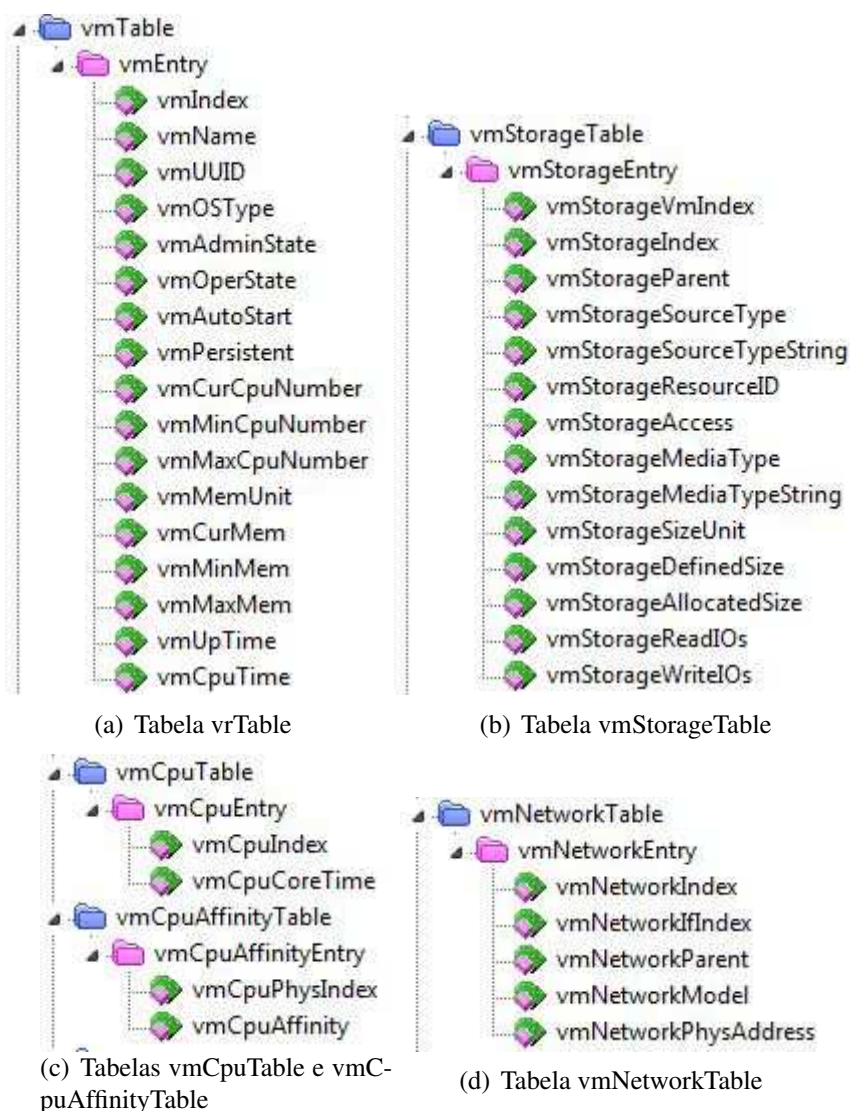


Figura 4. Tabelas da VMM-MIB

das MIBs supracitadas são diferentes, pois uma se refere a máquinas virtuais e a outra a roteadores virtuais. Felizmente, roteadores podem ser emulados por softwares, como por exemplo, o Vyatta [Vyatta 2013]. Sendo assim, máquinas virtuais executando o Vyatta podem ser consideradas, para todos os efeitos, um roteador virtual. Para isso, basta que o *hypervisor* seja executado dentro do roteador físico. Desse modo, não há problema em utilizar a VMM-MIB como base para uma MIB que possua as funcionalidades de gerenciamento de VRs desejadas.

Inspirada na VR-MIB, esta proposta acrescenta à VMM-MIB duas tabelas para configuração de VMs (ou VRs): a **vmConfigTable**, que permitirá a criação/remoção de VMs; e a **vmNetworkConfigTable**, que permitirá a criação/remoção de interfaces de rede virtuais e a associação com as físicas (*binding*). Além disso, três objetos escalares, para fins de controle, são incluídos: **vmConfigNextAvailableVmIndex**, **vmConfiguredVMs** e **vmActiveVMs**, com as mesmas funções dos objetos da VR-MIB **vrConfigNextAvailableVrId**, **vrConfiguredVRs** e **vrActiveVRs**, respectivamente. A Figura 6 mostra, em

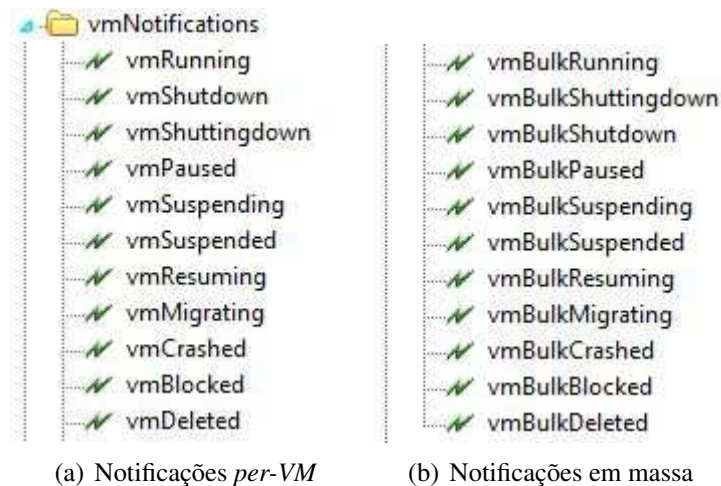


Figura 5. Objetos de notificação da VMM-MIB

destaque, as tabelas e os objetos escalares incluídos na nova VMM-MIB.

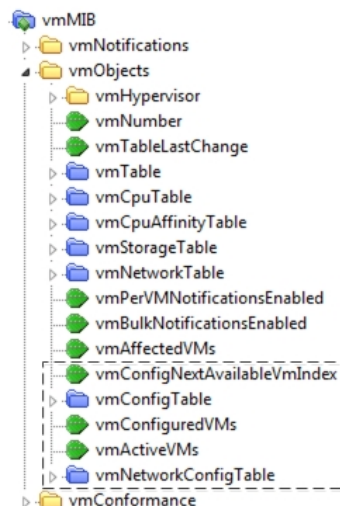


Figura 6. Objetos incluídos na nova VMM-MIB

A tabela **vmConfigTable** proposta contém objetos com as mesmas funcionalidades dos objetos da tabela **vrConfigTable** da VR-MIB. Além disso, nesta proposta, os objetos da tabela **vmTable** da VMM-MIB que possuíam permissões *read-write* são migrados para a **vmConfigTable**, pois esta tabela deve conter todos os objetos de configuração de uma nova VM. Nesta migração, as permissões de acesso *read-write* são trocadas para *read-create*, de forma a permitir a criação de novas entradas na tabela. Para evitar duplicação, os objetos **vmIndex** e **vmName** são retirados da tabela **vmTable**, posto que agora eles pertencem à **vmConfigTable**. A Figura 7 mostra a tabela **vmConfigTable** proposta e a origem de seus objetos.

A tabela **vmNetworkConfigTable** proposta contém objetos com as mesmas funcionalidades dos objetos da tabela **vrIfConfigTable** da VR-MIB. Só com esses objetos já é possível agregar as funcionalidades de criação e remoção de interfaces de rede virtuais, porém migrando-se os objetos **vmNetworkIfIndex** e **vmNetworkParent** da tabela

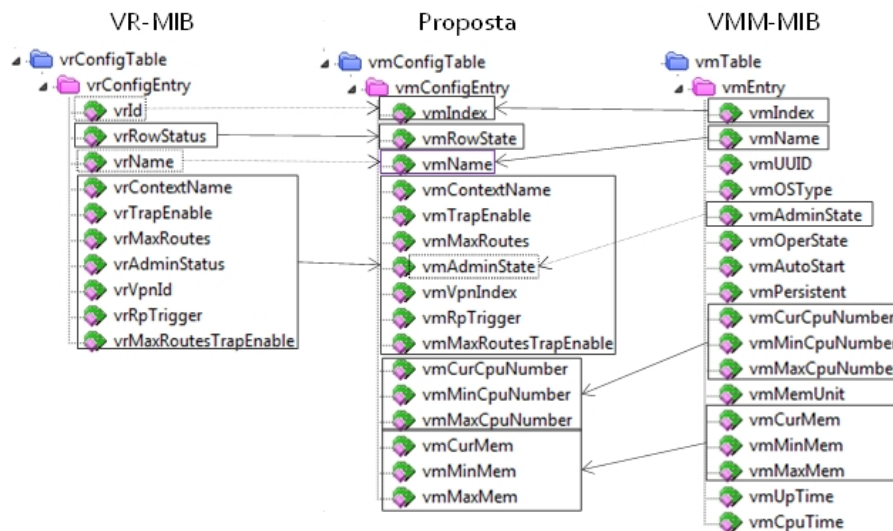


Figura 7. Objetos da tabela vmConfigTable proposta

vmNetworkTable da VMM-MIB para esta nova tabela, é possível associar a interface virtual à física (*binding*). Para permitir a criação de entradas na tabela, seus objetos têm permissão *read-create*. A Figura 8 mostra a tabela **vmNetworkConfigTable** proposta e a origem de seus objetos.

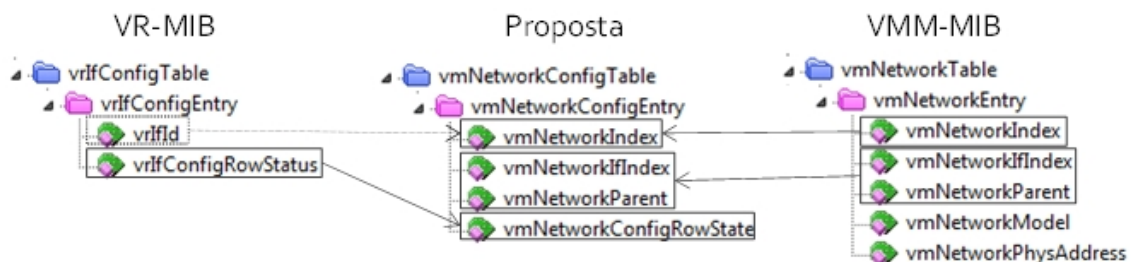


Figura 8. Objetos da tabela vmNetworkConfigTable proposta

Com a inclusão das tabelas **vmConfigTable** e **vmNetworkConfigTable** na VMM-MIB, já é possível obter as funcionalidades de configuração de alto-nível de roteadores virtuais oferecidas pela VR-MIB, porém, para abranger todas as informações estatísticas de roteadores virtuais, faz-se necessário, ainda, incluir na proposta objetos equivalentes aos da tabela **vrStatTable** da VR-MIB. Sendo assim, a tabela **vmTable** da nova VMM-MIB deve conter os objetos originais, exceto aqueles migrados para a tabela **vmConfigTable**, e os novos objetos **vmRouteEntries**, **vmFIBEntries**, **vmRpState**, **vmLoopbackAddressType** e **vmLoopbackAddress**, com as mesmas funções dos objetos da VR-MIB **vrStatRouteEntries**, **vrStatFIBEntries**, **vrRpStatus**, **vrRouterAddressType** e **vrRouterAddress**, respectivamente. A Figura 9 mostra a tabela **vmTable** proposta e a origem de seus novos objetos.

Com relação às notificações (*traps*), temos que a VMM-MIB possui uma grande quantidade de objetos referentes aos estados operacionais das VMs. Sendo assim, as notificações **vrUp** e **vrDown** da VR-MIB já são abrangidas pela VMM-MIB, ainda que de forma mais detalhada. O único objeto de notificação que necessita ser incluído na pro-

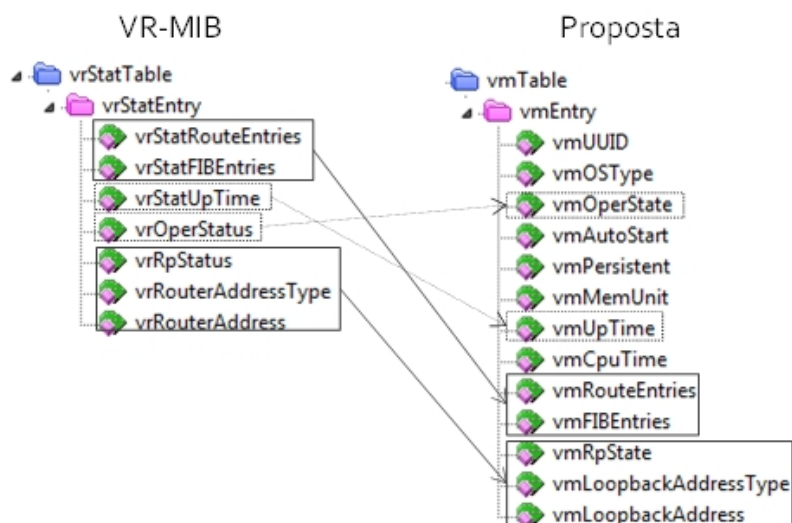


Figura 9. Objetos da tabela vmTable proposta

posta é o **vmMaxRoutesExceeded**, equivalente ao **vrMaxRoutesExceed** da VR-MIB, pois trata-se de uma notificação específica de roteadores, onde é informado que o número máximo de rotas foi excedido.

Nesta proposta, a tabela **vmNetworkTable** teve dois objetos migrados para a **vmNetworkConfigTable**, portanto agora, esta tabela tem apenas os objetos **vmNetworkModel** e **vmNetworkPhysAddress**, conforme mostrado na Figura 10. As tabelas **vmCpuTable**, **vmCpuAffinityTable** e **vmStorageTable** permanecem inalteradas.

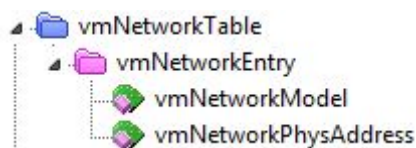


Figura 10. Objetos da tabela

4. Avaliação

Das diversas funcionalidades agregadas à VMM-MIB com as modificações propostas, a criação de VRs é a mais significativa, pois a mesma não é atualmente contemplada pela VMM-MIB original. Sendo assim, esta seção, apresenta a avaliação desta operação em uma interface de gerenciamento baseada no SNMP.

4.1. Interface de Gerenciamento

A interface de gerenciamento implementada utiliza a comunicação gerente-agente do SNMPv2, conforme ilustrado na Figura 11. Para a criação de um VR, o gerente SNMP envia uma mensagem **set-request** (T1) carregando as informações necessárias para o agente criar o VR (*i.e.*, **vmRowState**, **vmName**, **vmContextName**, **vmTrapEnable**, **vmAdminState**). Como reação, o agente emite uma requisição CLI (T2) ao *hypervisor* que efetivamente cria o roteador virtual (*e.g.*, `xe vm-install new-name-label=<vm-name> template=vyatta`, no caso do XenServer).

Como a ação do lado do roteador pode durar um tempo indefinido, o **set-request** é imediatamente respondido com uma mensagem **get-response** (T3). Quando a operação solicitada termina, uma mensagem de *trap* é emitida (T5), por um gerador de *traps* em execução dentro do roteador físico, informando que o roteador virtual foi criado com sucesso (T6).

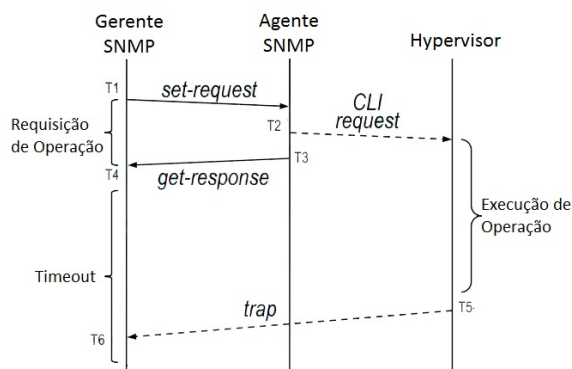


Figura 11. Fluxo de mensagens da interface de gerenciamento baseada no SNMP

4.2. Descrição do Ambiente

Os experimentos foram executados em um computador com processador Intel Core 2 Quad 2,4GHz com 4GB de memória RAM. Este computador foi configurado de modo a emular, para fins de avaliação, um roteador físico. A plataforma de virtualização (*hypervisor*) utilizada foi o Citrix XenServer 5.6 [XenServer 2013]. O *template* utilizado na criação de roteadores virtuais foi o Vyatta [Vyatta 2013].

O módulo VMM-MIB proposto, discutido na seção 3, foi compilado, instrumentado e adicionado ao agente disponibilizado no pacote NET-SNMP 5.5 (**snmpd**) [NET-SNMP 2013]. Operações SNMP são materializadas através de aplicações do pacote NET-SNMP correspondentes as mensagens do protocolo, *e.g.*, **snmpset**, **snmpget** e **snmpget-bulk**. Estas aplicações foram utilizadas para implementar o gerente SNMP. Além disso, a interface de gerenciamento foi avaliada utilizando-se a versão 2 do SNMP que é a versão mais popular do protocolo.

4.3. Resultados

O componente em estudo é a interface de gerenciamento descrita na seção 4.1, que tem como principal serviço a criação de VRs. Foram escolhidas duas métricas para a avaliação: tempo de resposta e consumo de recursos pelo agente SNMP (CPU e memória). O tempo de resposta reflete o tempo médio da criação de VRs na solução de gerenciamento proposta. A utilização de CPU e de memória RAM do roteador físico ilustra o impacto das operações de gerenciamento nos recursos do roteador físico.

A carga de trabalho consiste na criação de um VR adicional quando um número de VRs previamente criados está ativo (*i.e.*, em operação). O sistema começa sem nenhum VR e a cada nova requisição é criado um novo VR que permanece ativo, até que se atinja o total de 10 VRs ativos. A restrição de 10 VRs ativos se dá pelo fato de que esse foi o limite máximo suportado pela máquina utilizada nos testes.

Para esta avaliação foram conduzidos experimentos no ambiente de teste descrito, utilizando a medição como técnica de avaliação. Cada experimento foi repetido 30 ve-

zes sob as mesmas condições, a fim de se obter as médias populacionais com os seus respectivos intervalos de confiança. Para isto, foi utilizado um nível de confiança de 95%.

A Figura 12 ilustra o tempo de resposta (em segundos) na criação de 1 VR quando um número variável de VRs (0 até 10) está ativo.

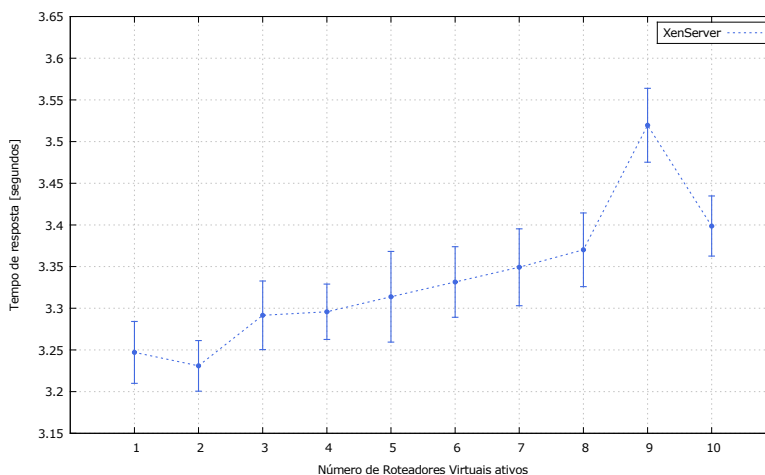


Figura 12. Criação de Roteador Virtual

É possível observar que, de uma maneira geral, o tempo de resposta cresce quando o número de VRs ativos aumenta. Isso acontece pelo fato de que os VRs ativos compartilham e consomem recursos da máquina física (CPU e memória), que estarão disponíveis em menor quantidade nas próximas requisições, o que influencia no aumento do tempo de resposta na criação de um novo VR. A exceção ocorre para quando 10 VRs estão ativos. O que ocorre aqui é que a máquina física não consegue suportar a carga de 10 VRs ativos simultaneamente (por falta de memória) e o hypervisor suspende alguns VRs arbitrariamente.

Em seguida, foi avaliado o consumo de CPU e memória pelo agente SNMP (snmpd). Verificou-se que o tempo de CPU do agente SNMP não apresenta mudanças significativas quando o número de VRs ativos aumenta até 10 VRs, permanecendo na ordem de 40 milissegundos, o que indica que o agente é capaz de armazenar dados de 10 VRs sem prejuízo significativo. A Figura 13 mostra o consumo de memória do agente SNMP na criação de VRs.

Os resultados da Figura 13 mostram o percentual de consumo de memória do agente SNMP em relação ao total disponível na máquina física quando um novo VR criado, variando o número de VRs ativos. Novamente, o consumo de memória do agente cresce quando o número de VRs ativo cresce. A primeira razão é que a quantidade de memória disponível na máquina física diminui a cada novo VR criado, e, conseqüentemente, a relação entre a memória requerida pra processar uma mesma mensagem **set-request** em relação ao total disponível também diminui. A segunda razão se deve ao fato de que o agente precisa armazenar uma quantidade de informações cada vez maior a cada novo VR, o que exige mais memória do agente SNMP.

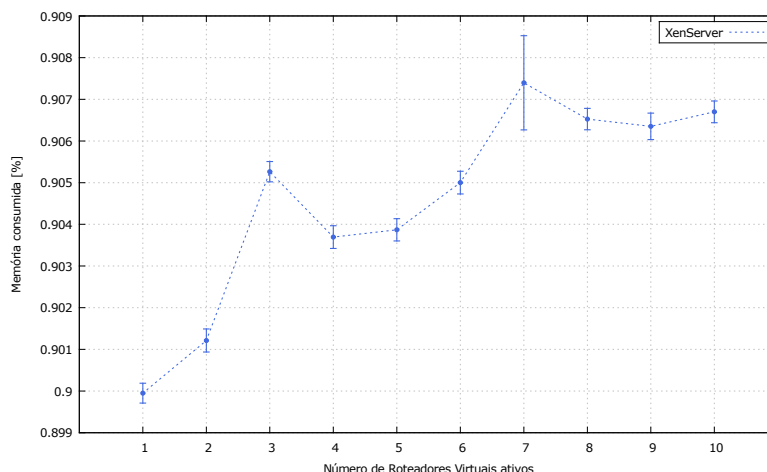


Figura 13. Consumo de Memória RAM

5. Conclusão

Neste trabalho foi apresentada uma proposta de modificação da VMM-MIB para tornar esta MIB funcional para o gerenciamento de roteadores virtuais em ambientes de virtualização de redes (NVEs). A VMM-MIB é uma proposta recente, que vem sendo atualizada desde 2012. Sendo assim, optou-se por agregar funcionalidades propostas para a VR-MIB à VMM-MIB e, assim, permitir a criação de roteadores virtuais.

Todos os objetos da **vrConfigTable** da VR-MIB foram incluídos na nova tabela **vmConfigTable** da VMM-MIB, com os nomes devidamente adaptados para o padrão da VMM-MIB, mas mantendo-se as permissões originais (*read-create*). Da mesma forma, todos os objetos da tabela **vrIfConfigTable** da VR-MIB foram adicionados à nova tabela **vmNetworkConfigTable** da VMM-MIB. Além disso, todos os objetos escalares da VR-MIB foram incluídos na nova VMM-MIB, de forma a permitir o controle dessas tabelas da mesma maneira que na VR-MIB. Para não perder as informações estatísticas dos roteadores virtuais criados, também foram incluídos na tabela **vmTable** da proposta todos os objetos da tabela **vrStatTable**, apesar de alguns deles já existirem na VMM-MIB original.

Uma interface de gerenciamento baseada nas extensões propostas para a VMM-MIB foi implementada e avaliada. Os resultados mostram que a memória é o recurso que exerce maior influência sobre o tempo de resposta de uma operação de criação de VRs. Quando o número de VRs aumenta, o agente SNMP consome mais memória, o que impacta no tempo necessário para a criação de novos VR. O consumo de CPU do agente SNMP não se mostrou significativo e não tem grande influência no tempo de resposta.

Como trabalhos futuros, pretende-se avaliar outras operações de gerenciamento como recuperação do estado de VRs, remoção e cópia de VRs, além de adaptar a VMM-MIB estendida para outros protocolos de gerenciamento como NETCONF e *Web services*.

Referências

Anderson, T., Peterson, L., Shenker, S., and Turner, J. (2005). Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41.

- Asai, H., MacFaden, M., Schoenwaelder, J., Sekiya, Y., Shima, K., Tsou, T., Zhou, C., and Esaki, H. (2013). Management Information Base for Virtual Machines Controlled by a Hypervisor. <http://tools.ietf.org/html/draft-asai-vmm-mib-05>. Internet draft.
- Chowdhury, N. M. M. K. and Boutaba, R. (2009). Network virtualization: state of the art and research challenges. *Communications Magazine*, 47(7):20–26.
- Chowdhury, N. M. M. K. and Boutaba, R. (2010). A survey of network virtualization. *Computer Network*, 54(5):862–876.
- Daitx, F., Esteves, R., and Granville, L. (2011). On the use of SNMP as a management interface for virtual networks. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 177–184.
- Esteves, R., Granville, L., and Boutaba, R. (2013). On the management of virtual networks. *Communications Magazine, IEEE*, 51(7).
- McCloghrie, K. and Kastenholz, F. (2000). The Interfaces Group MIB. RFC 2863 (Draft Standard).
- NET-SNMP (2013). <http://www.net-snmp.org/>. Acessado em novembro de 2013.
- Stelzer, E., Hancock, S., Schliesser, B., and Laria, J. (2003). Virtual Router Management Information Base Using SMIV2. <http://tools.ietf.org/html/draft-ietf-ppvpn-vr-mib-05>. Internet draft.
- Stelzer, E., Hancock, S., Schliesser, B., and Laria, J. (2005). Virtual Router Management Information Base Using SMIV2. <http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04>. Internet draft.
- Vyatta (2013). Vyatta.org community. <http://www.vyatta.org/>. Acessado em outubro de 2013.
- XenServer (2013). Open source virtualization. <http://www.xenserver.org/>. Acessado em outubro de 2013.

ANEXO B ARTIGO PUBLICADO – IM 2015

Este artigo propõe um modelo de dados atualizado como alternativa à VR-MIB, considerando que esta não progrediu no caminho da padronização. A MIB proposta, chamada NEW-VMM-MIB, foi inspirada na VMM-MIB e está alinhada com os atuais esforços do IETF para o gerenciamento de ambientes virtualizados. Além disso, este artigo propõe e investiga quatro interfaces de gerenciamento baseadas em SNMPv2c, SNMPv3, NETCONF, e RESTful Web Services, respectivamente, para o gerenciamento de roteadores físicos que suportam a virtualização. As interfaces implementadas são comparadas através da avaliação do desempenho de cada uma delas em termos do tempo de resposta, do tempo de CPU, do consumo de memória e do uso da rede, para três operações básicas de gerenciamento de VRs: criação, recuperação de informações e remoção.

- **Título:**

Evaluating SNMP, NETCONF, and RESTful Web Services for Router Virtualization Management

- **Conferência:**

14th IFIP/IEEE *International Symposium on Integrated Network Management* (IM - 2015)

- **Tipo:**

Trilha principal (*full-paper*)

- **Qualis:**

B1

- **URL:**

<<http://dx.doi.org/10.1109/INM.2015.7140284>>

- **Data:**

11 a 15 de maio de 2015

- **Realizado em:**

Ottawa, Canada

Evaluating SNMP, NETCONF, and RESTful Web Services for Router Virtualization Management

Paulo Roberto da Paz Ferraz Santos, Rafael Pereira Esteves,
 Lisandro Zambenedetti Granville
 Federal University of Rio Grande do Sul, Porto Alegre, Brazil
 Email: {prpfsantos, rpesteves, granville}@inf.ufrgs.br

Abstract— In network virtualization environments (NVEs), the physical infrastructure is shared among different users (or service providers) who create multiple virtual networks (VNs). As part of VN provisioning, virtual routers (VRs) are created inside physical routers supporting virtualization. Currently, the management of NVEs is mostly realized by proprietary solutions. Heterogeneous NVEs (*i.e.*, with different equipment and technologies) are difficult to manage due to the lack of standardized management solutions. As a first step to achieve management interoperability, good performance, and high scalability, we implemented, evaluated, and compared four management interfaces for physical routers that host virtual ones. The interfaces are based on SNMP (v2c and v3), NETCONF, and RESTful Web Services, and are designed to perform three basic VR management operations: VR creation, VR retrieval, and VR removal. We evaluate these interfaces with regard to the following metrics: response time, CPU time, memory consumption, and network usage. Results show that the SNMPv2c interface is the most suitable one for small NVEs without strict security requirements and NETCONF is the best choice to compose a management interface to be deployed in more realistic scenarios, where security and scalability are major concerns.

I. INTRODUCTION

Network virtualization emerged as an alternative to support the development of new network architectures and overcome Internet ossification [1]. In a network virtualization environment (NVE), the underlying physical infrastructure, called substrate, is shared among different users (or service providers) who create multiple virtual networks (VNs), each one employing independent protocols, forwarding schemes, and policies. 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 [2]. Thus, virtual routers (VRs) are created and hosted within physical routers, enabling multiple independent networks to operate simultaneously on a single physical infrastructure [3].

Current NVEs are mostly manually operated through proprietary command line interfaces (CLIs) or proprietary management tools. The management of NVEs built on top of heterogeneous physical substrates (*i.e.*, with different equipment and technologies) poses difficulties to NVE operators because several independent tools are required to perform a management task, *e.g.*, creating a virtual network. To minimize the hurdle of managing a diversified pool of physical resources, standardized management interfaces must be defined and evaluated to allow interoperability between different virtualization platforms and management tools. In addition, virtual networks may be constructed from resources located

at different administrative domains [4] and must be properly managed. Therefore, NVE operators should opt for a management interface/protocol that allows efficient and scalable management of physical routers hosting several virtual ones.

Standardization efforts have been conducted under Internet Engineering Task Force (IETF) to enable the management of VRs in different contexts. For example, the VR-MIB [5] was proposed to manage VRs in the context of L3VPNs. The VRRP-MIB [6] defines a VR as an abstract representation of a physical router and is used to handle failures. Motivated by the IETF efforts, Daitx *et al.* [2] proposed an NVE management solution based on Simple Network Management Protocol version 2c (SNMPv2c) extending the VR-MIB [5]. Although SNMP is not traditionally used for configuration-related tasks, Daitx *et al.* have demonstrated that SNMP can still be used in NVE management.

In this article, we evaluate SNMP version 3 (SNMPv3) [7], Network Configuration Protocol (NETCONF) [8], and RESTful Web services (RWS) ¹ as alternatives to SNMPv2c for the management of physical routers supporting virtualization, thus extending the analysis presented by Daitx *et al.* [2]. SNMPv3 has been investigated because of its security and remote configuration features [9] that were not originally present in SNMPv2c. NETCONF and RWS have been considered possible alternatives to SNMP for network management and can overcome the limitations of SNMP in terms of scalability [10], [11]. However, neither NETCONF or RWS have been properly investigated in the context of router virtualization and management issues related to the NVE domain require further analysis. RWS were chosen because they are more lightweight than SOAP-based Web services and are being widely deployed in many Web services implementations.

In addition, we propose an updated data model as alternative to VR-MIB considering that VR-MIB did not progress in the standardization path, remaining as an expired draft since 2006 and leaving the area with no SNMP-based solution. Inspired by the recent VMM-MIB [12], we proposed a new MIB, called NEW-VMM-MIB, which is aligned with current IETF efforts towards the management of virtualized environments. For each protocol, we developed a corresponding management interface and compared them using three basic operations: VR creation, VR retrieval, and VR removal. We evaluated these interfaces with regard to the following metrics: response time, CPU time, memory consumption, and network usage.

¹From this point we use the acronym RWS to refer to RESTful Web services along the text

In our implementations we used Net-SNMP [13] for SNMP, OpenYUMA [14] for NETCONF, and Jersey [15] for RWS. We used the XenServer [16] hypervisor to emulate a physical router supporting virtualization and the Vyatta routing suite [17] to implement VRs.

The remaining of this article is organized as follows. In Section II, we review both VR-MIB and VMM-MIB proposals along with other solutions that rely on the aforementioned protocols in network virtualization management. In Section III, we outline the architecture of a manageable physical router hosting virtual ones, describe the updated data model, and explain the message flow of each proposed management interface. The performance of the proposed management interfaces is evaluated and the results achieved are presented in Section IV. Finally, in Section V we conclude the article with final remarks and directions for future work.

II. RELATED WORK

Network virtualization reduces infrastructure costs by allowing multiple virtual resources to share a single physical one, which is often referred to as network consolidation [18]. Moreover, network virtualization improves the reliability of network infrastructures by offering hardware independence to virtual network components (*e.g.*, virtual routers). Management is considered a crucial aspect to enable NVEs, which is endorsed by many research projects that consider NVE management at the design stage, instead of tackling it after the NVE is already operational [19]. In this section, we discuss a number of proposals related to the management of NVEs.

The Virtual Router MIB (VR-MIB) [5], originally proposed by the L3VPN IETF Work Group [20] for Layer 3 Virtual Private Networks (L3VPNs), was one of the first initiatives to use SNMP for router virtualization management. The VR-MIB module contains objects related to the high-level configuration of VRs structured in three main tables, as shown in Fig.1(a): the `vrConfigTable`, responsible for the creation and removal of VRs; the `vrStatTable`, which stores statistics of VRs hosted in a physical router; and the `vrIfConfigTable` that manages the mapping between VRs and their virtual network interfaces. The main problem with VR-MIB is the lack of progression in the IETF standardization path and the absence of updates since 2006.

In July 2012, the VMM-MIB [12] was proposed in IETF for managing virtual machines (VMs) controlled by a hypervisor. This MIB module, which the structure is shown in Fig.1(b), consists of managed objects related to the hypervisor (`vmHypervisor` group), VMs (`vmTable`), and virtual resources allocated to VMs, *i.e.*, processors (`vmCpuTable`, `vmCpuAffinityTable`), network interfaces (`vmNetworkTable`), and storage devices (`vmStorageTable`). Despite the VMM-MIB appears to be useful for the management of VRs, because a VR can be implemented by a software-based router running in a VM, most objects of this MIB module are `read-only`, preventing proper VR configuration. Therefore, VMM-MIB requires configurable objects to support, for example, VR creation and removal.

Daitx *et al.* [2] have proposed a network virtualization management interface based on SNMP, extending the VR-MIB module to allow flexible interface binding. Daitx *et al.*

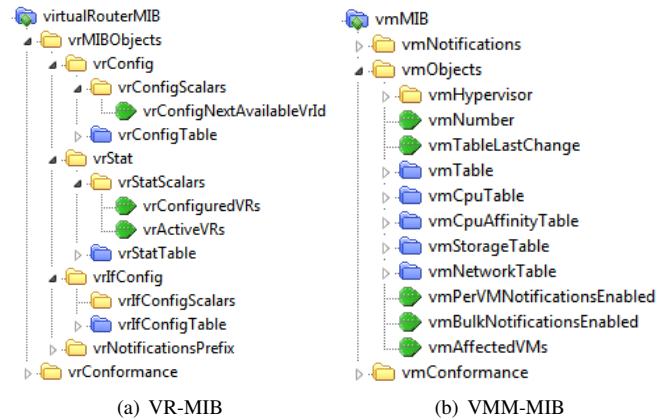


Fig. 1. VR-MIB and VMM-MIB structures

used some basic management operations to evaluate the performance of the proposed interface in two virtualization platforms: XenServer and VMWare. Although, they have shown that SNMP performance largely depends on the virtualization platform, they did not investigate others management protocols and used the obsolete VR-MIB as data model.

Patricio *et al.* [21] have proposed a NETCONF-based interface for configuring virtual switches in VLAN-based virtual networks. This interface has been tested with Open vSwitch (OVS) [22] in the context of Software Defined Networks (SDNs) and supported five management operations: creation and removal of virtual switches, creation and removal of a port of a switch, and creation of a port with a VLAN tag in a switch. Aside from the fact that this interface did not manage virtual routers but switches, there was no performance evaluation to demonstrate the feasibility of the proposed interface.

Rendon *et al.* [23] have used the Mashup [24] [25] technology to provide a mechanism able to monitor heterogeneous Virtual Nodes. The proposed model enables any Virtual Infrastructure Administrator to adapt, customize, and combine existing monitoring tools in order to improve system and network monitoring tasks in virtualized environments. Rendon *et al.* have used RWS to develop Virtual Node Wrappers. These elements, located at the Adaption Layer, receive service requests from a Composition Layer and provide monitoring operations to a Managed Resources Layer to hide the complexity and heterogeneity of the substrate. Although this work has proposed a flexible and extensible system, it focused on monitoring and did not support configuration of Virtual Nodes (*e.g.*, VR creation).

The previous works addressed several management issues and provided important contributions to the network virtualization area. However, most proposals have not evaluated traditional network management protocols when applied to router virtualization management. This investigation is important to provide a better understanding about the benefits and limitations of such protocols in NVE management and identify requirements of a standardized management interface for physical routers supporting virtualization. Vendors will be encouraged to support such an interface to allow their products to be easily compatible with existing management tools and thereby facilitate the deployment of their equipment in NVEs

with predominance of other manufacturers.

III. MANAGEMENT INTERFACES FOR ROUTER VIRTUALIZATION

In this section we first introduce the conceptual architecture of a physical router hosting virtual ones [2]. Then, we present the proposed management interfaces for SNMPv2c, SNMPv3, NETCONF, and RWS, respectively, describing the data model and the message flow for each management interface.

A. Architecture

A physical router is conceptually divided in two layers: a substrate layer and a virtualization layer. The substrate layer consists of the physical device itself and a layer of software, called hypervisor, that allows deployment of VRs on top of it. The virtualization layer comprises multiple isolated VRs running their own control planes. In our proposed architecture, depicted in Fig.2, the management interface is located between the VRs and the hypervisor.

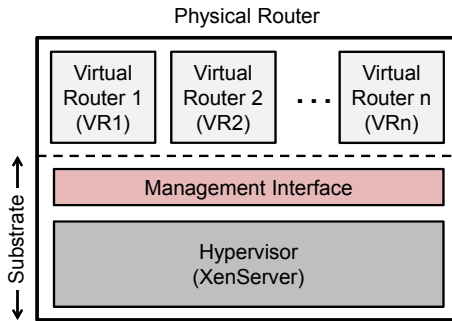


Fig. 2. Architecture of a physical router hosting VRs

The management interface allows the Infrastructure Provider (InP) operator to access, operate, maintain, and administer a physical router that supports virtualization [19]. In our case, the InP operator uses the management interface to perform basic VR management operations, such as creation, removal, and information retrieval. The definition of an appropriate data model is fundamental to support such operations.

B. Data Model

As a starting point to define a data model, we analyzed the VR-MIB structure, objects, and functionality. As mentioned before, VR-MIB is outdated, which hinders its utilization in practice. On the other hand, the VMM-MIB is an ongoing effort that focus on the management of virtual machines controlled by a hypervisor. However, VMM-MIB currently does not support VR configuration. To address these issues, we propose a NEW-VMM-MIB using the VMM-MIB as a basis and incorporating objects from VR-MIB to enable proper VR management. To achieve this, we had to add tables and objects from VR-MIB, changed access permissions for some objects in VMM-MIB, and migrated some objects from their original tables to other tables.

As depicted in Fig.3, the NEW-VMM-MIB has the same basic structure of VMM-MIB, with the addition of 2 new tables (`vmConfigTable` and `vmNetworkConfigTable`) and

3 new scalar objects (`vmConfigNextAvailableVmIndex`², `vmConfiguredVMs`³, and `vmActiveVMs`⁴) adapted from VR-MIB. The `vmConfigTable`, depicted in Fig.3(c), has been included in NEW-VMM-MIB containing objects with *read-create* permission, inspired by VR-MIB, to allow VR configuration features that were missing in VMM-MIB, e.g., VR creation and removal. The `vmNetworkConfigTable`, depicted in Fig.3(d), has also been included to enable the configuration of VR network interfaces. VR routing information is modeled by 5 objects (Fig.3(b)) added to `vmTable`, based on `vrStatTable` of VR-MIB. Moreover, the objects `vmName` and `vmAdminState` have been migrated from `vmTable` to `vmConfigTable`. As a result, the NEW-VMM-MIB is capable of managing VRs and VMs as well, and consequently, we used it as the data model of the SNMP management interface.

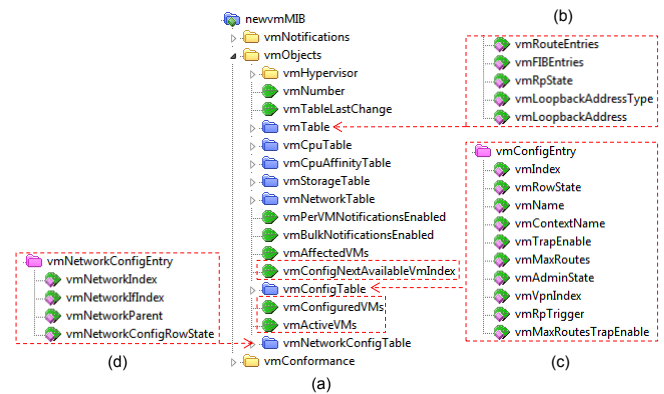


Fig. 3. NEW-VMM-MIB structure with new objects and tables

YANG is a data modeling language used to model device configuration, device state, remote procedure calls, and notifications using the NETCONF protocol [26] and we can translate Structure of Management Information version 2 (SMIv2) MIB modules to YANG modules [27]. Thus, we have defined an YANG-based data model for the NETCONF interface by directly translating the NEW-VMM-MIB module. The tree structure remains the same of NEW-VMM-MIB but *tables* are translated into *lists* and *objects* into *leaves*.

Unlike SNMP and NETCONF, the RWS solution does not have a standardized data model language. In RWS, the data model is highly coupled with the implementation. Because we have implemented a Java solution for the RWS interface, we used the Java Architecture for XML Binding (JAXB) [28] to create the data model. Using JAXB, each row of a table (e.g., `vmConfigTable`) is represented in XML format but is stored as an *Object* in a `Map<K, V>`⁵, where the index (e.g., `vmIndex`) refers to the *key*.

Despite the particularities of each data model, the information that is modeled is essentially the same. The key difference between the proposed management interfaces relies

²`vmConfigNextAvailableVmIndex` provides the next available VM index value to be used to create an entry in the associated `vmConfigTable`

³`vmConfiguredVMs` provides the number VMs configured in the network element.

⁴`vmActiveVMs` provides the number of active VMs in the network element.

⁵`Map<K, V>` is an object that maps a key (K) to one value (V) at most and cannot contain duplicate keys.

on the management protocol, *i.e.*, SNMP, NETCONF, and RWS. Next, the messages exchanged between a manager (or client) and an agent (or server) for each interface (*i.e.*, the message flow) are discussed.

C. Message Flow

The message flow for VR creation or VR removal in the SNMPv2, SNMPv3, NETCONF, and RWS management interfaces is shown in Fig.4. The message flow for VR information retrieval for each management interfaces is shown in Fig.5. We start by describing the message flow of the SNMP interface.

A typical message exchange between an SNMPv2c manager and an agent using the proposed NEW-VM-MIB module to create/remove VR is depicted in Fig.4(a). Initially, the manager sends an SNMP `set-request` carrying the information needed by the agent to perform the desired operation. As a reaction, the agent issues a CLI request to the hypervisor that actually creates/removes the VR. Because the action at the router side may last awhile, the `set-request` is immediately replied in parallel with a `get-response`. When the requested operation finishes, a `trap` message is issued by a trap generator running inside the physical router, informing that the VR was successfully created/deleted. Since the trap message can be lost (because SNMP uses UDP), the manager initiates a timer together with the first `set-request`. If the timer expires and the confirmation trap is not received, the manager cannot guarantee the VR creation/removal. In this case, additional queries to the agent are needed to confirm the operation.

To retrieve information of a VR, the manager sends an SNMP `getbulk-request` carrying the OID of the required objects. The manager must know the `vmIndex` to retrieve information of a specific VR. Otherwise, the manager can discover the VR index by consulting the `vmConfigTable` and matching another attribute, *e.g.*, the VR name (`vmName`). The agent replies with a `get-response` containing the values of the requested objects, as depicted in Fig.5(a).

The SNMPv3 defines a number of security-related capabilities and, among them, the User-based Security Model (USM) [29] is largely used. USM provides authentication and privacy (encryption) services for SNMP. We choose HMAC-SHA-96 as the authentication protocol [29] and AES encryption [30]. Initially, the manager sends an SNMP `get-request` requesting `contextEngineID`⁶ and `contextName`⁷ [31]. Then, the agent responds with a `report PDU` message carrying, in addition to context information, security-related parameters in the message header. Subsequent packets contain the same messages exchanged by the SNMPv2 protocol, but are encrypted by AES and have USM security parameters in the header. The creation/removal operation messages are shown in Fig.4(b) and the retrieval ones in Fig.5(b).

In the NETCONF management interface, we choose SSH as the transport mechanism [32]. For the sake of simplicity, SSH is not depicted. In order to perform a configuration action, the NETCONF client (manager) establishes a session with the server (agent) only if a previous session is not present.

⁶contextEngineID uniquely identifies an SNMP entity that may realize an instance of a context with a particular contextName.

⁷contextName must be unique within an SNMP entity and is used to name a context.

The session establishment involves opening a TCP connection (omitted in the diagrams for simplicity) and the subsequent exchange of `<hello>` messages. After session establishment, the NETCONF client issues an `<edit-config>` message containing VR data (in XML format). After a `<commit>`, this configuration is applied to the *running* datastore triggering the hypervisor to actually create or remove the VR, as shown in Fig.4(c). The action of creating or removing a VR is defined in XML data as `operation="create"` or `operation="delete"`, respectively.

As depicted in Fig.5(c), to retrieve information of a VR, the client must send a `<get-config>` message to the NETCONF server. The server responds with a `<rpc-reply>` message if the operation is successful, or with a `<rpc-error>` message otherwise. If the retrieval operation is successful, the response message contains a `<data>` element in the query results. After all configurations/queries, the NETCONF session can be closed by sending the `<close-session>` message to the NETCONF server.

The RWS management interface, as the name says, is based on REST architecture [33]. The RWS solution uses HTTP methods (*e.g.*, GET, POST and DELETE) as verbs to request a resource identified by a Uniform Resource Identifier (URI) [34], [35]. In order to create a VR, the RWS client sends a POST method with data in HTTP, XML, or JSON formats. We chose XML for fairness because NETCONF data is also based on XML. The RWS server processes the request, stores the new VR information, and calls the hypervisor to create the VR. After successfully creating the VR, the server answers by sending a HTTP response with status code 200 OK [36], as shown in Fig.4(d). VR removal (Fig.4(d)) is analogous to VR creation, except for the fact that the client sends a DELETE method to an URI containing the index of the VR (`vmIndex`) to be removed.

The VR information retrieval is just as simple as VR creation and removal. As shown in Fig.5(d), the client sends a GET method to a URI containing the index of the VR to be queried. The server searches in the datastore and, in case of success, sends back a HTTP response containing the retrieved data in XML format. If the queried VR index does not exist in the datastore, the server responds with status code 204 No Content.

IV. EVALUATION

In this section we evaluated management interfaces based on SNMPv2, SNMPv3, NETCONF, and RWS to investigate which is the best solution for managing physical routers hosting VRs with regard to performance and scalability. We start by describing the test environment and pointing out all the tools used in the experiments. Then, we explain the methodology and define the metrics used in the evaluation. Finally, we present and analyze the results.

A. Test Environment

The experiments were performed on a computer with Intel Core 2 Duo 1.86GHz processor and 4GB of RAM. Citrix XenServer 5.6 [16] was used to emulate a physical router supporting virtualization. Vyatta 6.2 [17] was used as a template for VRs.

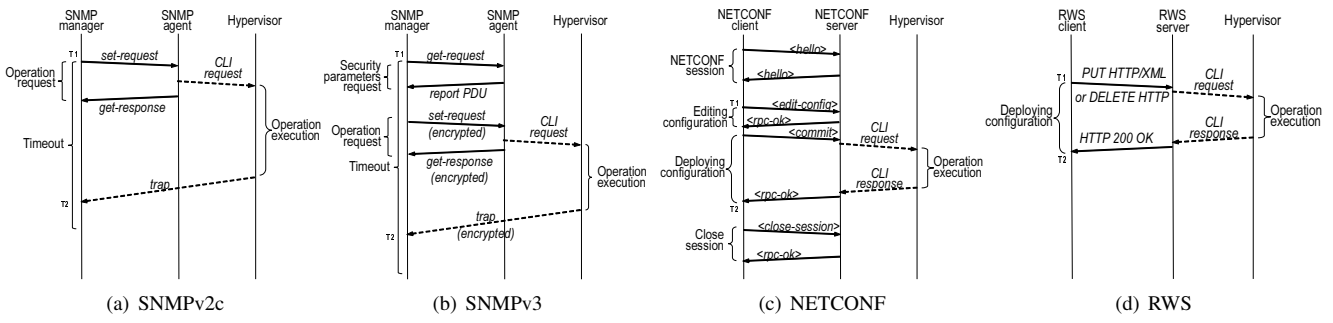


Fig. 4. Message flow of VR creation or VR removal operations

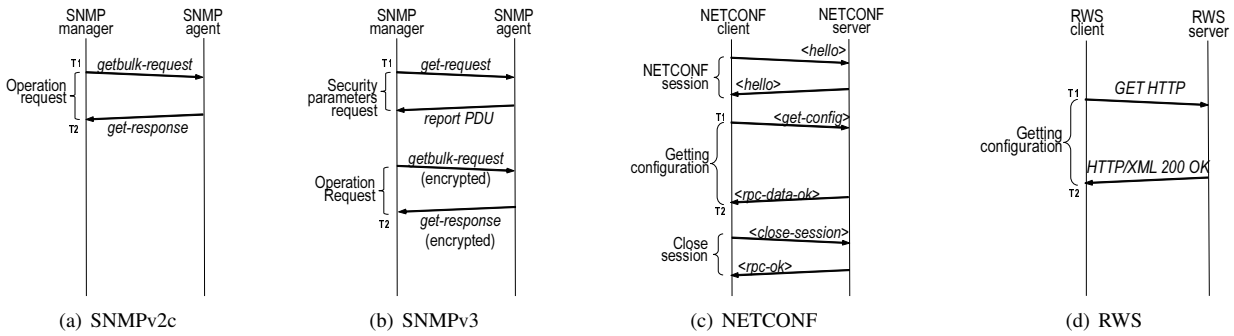


Fig. 5. Message flow of VR retrieval operation

The proposed NEW-VMM-MIB module, previously discussed in Section III-B, has been compiled, instrumented, and attached to a Net-SNMP 5.7.2 agent (*snmpd*) [13]. The SNMP interface runs SNMP versions 2c and 3. The SNMP interface uses two Net-SNMP applications: *snmpset* to send SetRequest messages and *snmpgetbulk* to send GetBulkRequest messages. The SNMPv3 interface has been configured to use SHA as the authentication protocol and AES for data encryption.

We implemented the NETCONF interface using the OpenYUMA (YANG Unified Modular Automation) 2.2.5 toolkit [14]. OpenYUMA includes a NETCONF client and server (*yangcli* and *netconfd*), a YANG module compiler (*yangdump*), and a server instrumentation library (*SIL*). We translated the NEW-VMM-MIB from SMIV2 to YANG, according to RFC6643 [27], and compiled the new YANG module using *yangdump*. Then, we instrumented the generated *SIL* code and attached it to a NETCONF agent. We also have configured this interface to use NETCONF over SSH mapping.

Finally, we deployed the RWS interface using Jersey [15] implementation of JAX-RS [37], a Java API for RESTful Web services. We have used Apache Tomcat 8.0.11 [38] with Java SE Runtime Environment 1.8.0_20 [39] as a server to run the Jersey servlet, and cURL 7.37.1 [40] as a client to send messages to a URL using the HTTP methods POST, GET, or DELETE.

As a common characteristic, all implementations have been integrated into the virtualization platform through the native management interface of XenServer, *xe* CLI.

B. Methodology and Metrics

The experiments have been conducted according to the following sequence of operations: create VR, start VR, retrieve information about the created VR, and, finally, remove VR. Each step (except for ‘start VR’) was a management operation measured with regard to response time, CPU time, memory consumption, and network usage.

The response time is the total time that each VR operation takes to be completed. This evaluation intends to discover if there are significant differences between the interfaces regarding response time. Here, we consider response time as the interval between the first (T_1) and the last (T_2) timestamps, indicated in Fig.4 and 5. All management operations were performed locally, *i.e.*, managers (clients) and agents (servers) were executed in the same computer, to avoid network delays.

CPU time and memory consumption metrics check the processing time and the amount of memory used by the server/agent when performing a management operation, *e.g.*, VR creation. Both metrics were obtained using the `ps` program from *procps* package [41] to gather CPU time and memory consumption of the server/agent processes, *i.e.*, *snmpd* (SNMP), *netconfd* (NETCONF), and *java* (RWS). CPU time and memory consumption were collected at the beginning (T_1) and at the end (T_2) of every manager operation, as indicated in Fig.4 and 5. CPU time is calculated as the difference between the final CPU time and the initial one, and memory consumption is the last memory measurement returned by `ps` because it reflects a percentage of the host total memory. In XenServer, the host memory is limited by

the memory of *Domain-0* ($dom0^8$). Therefore, we calculated memory consumption by multiplying the percentage of used memory (in decimals) by the total memory of $dom0$.

After 30 cycles of measures, we increased the number of pre-existing active VRs. Each cycle is a sequence of management operations for a single VR, *i.e.*, creation, start, retrieval, and removal. In order to obtain statistically sound results, we used a confidence level of 95%.

The last evaluated metric reflects the network usage when management traffic is generated to create, retrieve, and remove VRs. The goal of this evaluation is to verify the impact of each management interface on the network. Network usage was evaluated using `tcpdump` [42] program to capture network packets exchanged between clients and servers (*i.e.*, managers and agents), including protocol overhead and session establishment/termination. Then, we calculated the amount of traffic (in bytes), by summing the length of exchanged packets for each type of operation, *i.e.*, creation, retrieval, and removal. As in the previous experiment, this one was performed locally to avoid other network-related issues, such as packet loss.

Unlike the previous experiments, for network usage we increased the number of VRs to be created, retrieved, or removed after 30 cycles of measure, instead of increasing the number of pre-existing active VRs. Network usage results were also presented with a confidence level of 95%.

C. Results

As a first observation, we found out that our XenServer did not allowed more than 6 active VRs simultaneously. This happens because our XenServer allocates 752MB of RAM for *dom0* (by default) and each Vyatta-based VR allocates 512MB (the total RAM is 4GB). Using the `xentop` utility we could get information about the memory allocated to *dom0* and to the active VRs. Fig.6 shows the total memory consumption in our setup when VRs are activated.

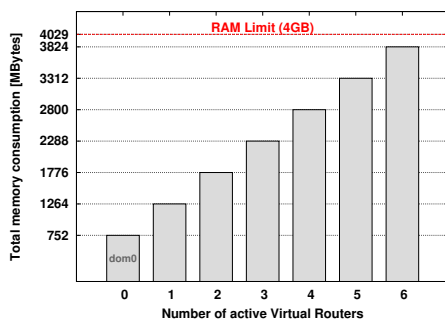


Fig. 6. Memory consumption of our XenServer and active virtual routers

Regarding response time in VR creation, depicted in Fig.7(a), the SNMPv2c and RWS interfaces presented the lowest response times, with statistically equal results. NETCONF and SNMPv3 also presented similar results but slightly higher

⁸*Domain-0*, *dom0*, or Domain Zero is the initial domain started by the Xen hypervisor on boot and contains the host operating system.

than the previous two. For VR retrieval, the SNMPv2c interface presented a faster response, followed by NETCONF, SNMPv3, and RWS, as shown in Fig.7(b). For VR removal, SNMPv2c interface presented lower response time than the other three interfaces, whose results were statistically equal in most measurements, as shown in Fig.7(c). However, VR removal presented a far greater response time than VR creation because XenServer has to shut down a VR before deleting it, and this process takes a considerable time.

Fig.8 shows that CPU time varied irregularly as we have increased the workload. In spite of these variations, the RWS interface presented higher CPU time than the others, showing that the Java process needs more CPU compared to the others server/agents processes. The SNMPv2c interface presented the lowest CPU time in most measurements for VR creation and removal (Fig.8(a) and Fig.8(c)). For VR retrieval, the NETCONF interface resulted in the lowest CPU time when the number of pre-existing active VRs is two or higher, as shown in Fig.8(b).

With respect to memory consumption, Fig.9 shows that the RWS solution consumes much more (around 70MB) memory than the others. This result was expected because the Java process is more robust and heavier than both `snmpd` and `netconfd` processes. The SNMPv2c interface had the lowest memory consumption, followed closely by SNMPv3 and NETCONF. Memory consumption presented slight linear growth for SNMPv2c, SNMPv3, and NETCONF as the workload increased. However, results for RWS showed some variation.

The experiments for network usage were conducted without VR activation because VR activation does not generate network traffic. Therefore, it was possible to evaluate network usage for more than 6 VRs. VR configuration results (Fig.10(a) and 10(c)) show that for few VRs (up to 16 for creation, 20 for removal), the NETCONF interface generated the highest amount of traffic, followed by RWS, SNMPv3, and SNMPv2c. When the number of managed VRs was increased, NETCONF traffic grew less compared to the other interfaces. NETCONF became better than RWS when 16 or more VRs were created (Fig.10(a)) or 20 or more VRs were removed (Fig.10(c)). When 22 VRs were created or 30 or more VRs were removed, NETCONF became better than SNMPv3. Regarding retrieval operations, Fig.10(b) shows that network traffic grows uniformly when the number of retrieved VRs increases for all interfaces. However, the NETCONF interface generated more traffic compared to the others. The SNMPv2c interface presented the lowest network usage for all operations.

The network usage results reflects protocol overhead. SNMPv2c uses UDP and therefore has the lowest overhead. SNMPv3, despite using UDP as well, is configured to use authentication and encryption, which results in a higher overhead than version 2. RWS is based on HTTP, which uses TCP, and therefore has higher overhead than SNMP. NETCONF has the highest overhead, because it uses an XML-based data encoding for both configuration data and protocol messages, and uses SSH as the transport mechanism. In spite of that, the NETCONF interface presents a lower traffic growth when the number of created/removed VRs increases. This can be explained by the fact that NETCONF first opens the session,

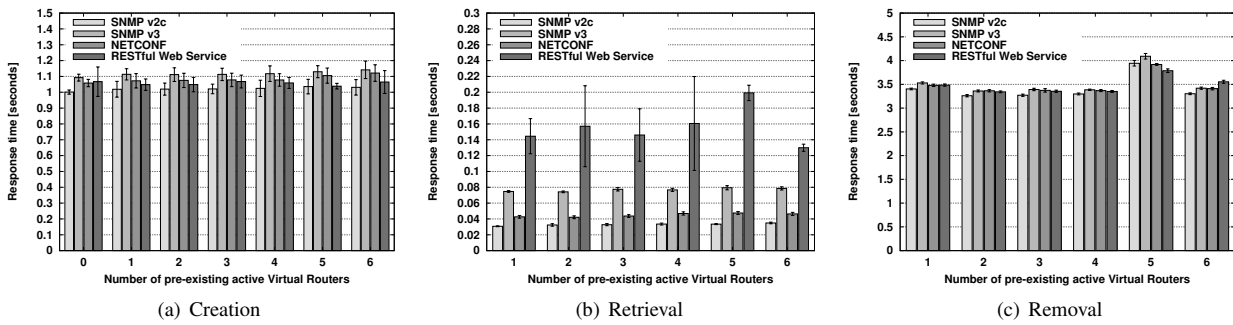


Fig. 7. Response time of the management interfaces

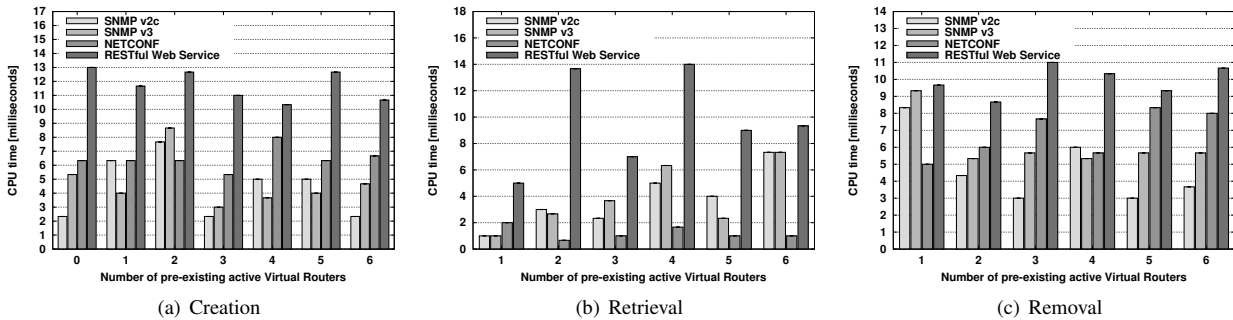


Fig. 8. CPU time of the management interfaces

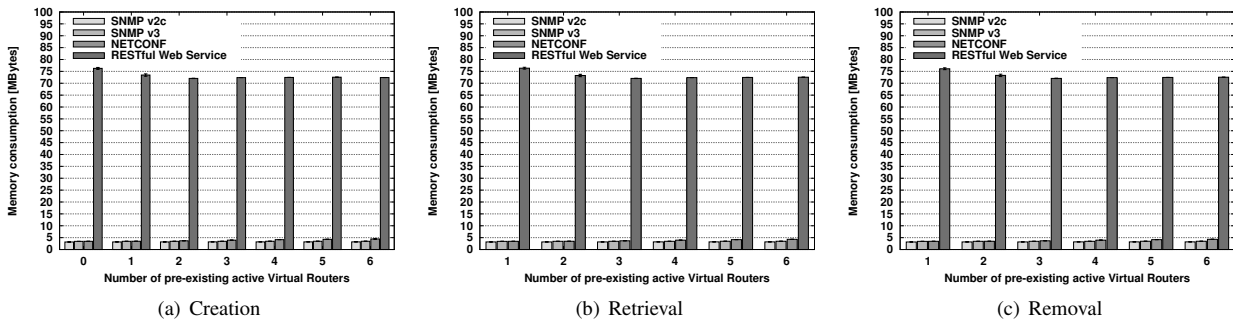


Fig. 9. Memory consumption of the management interfaces

deploys all VRs configurations and, at last, closes the session. In contrast, the interfaces based on SNMP and RWS configure multiple VRs by configuring each VR independently. On the other hand, for VR retrieval, SNMP and RWS are more efficient because SNMP relies on `GetBulkRequest` messages to get all objects at once and RWS retrieves all objects stored in the `Map` with a single HTTP GET message.

D. Comparison

Regarding response time, the SNMPv2c interface presented the best results. Considering the similarity of the results, we analyzed the average response time of the management interfaces, as shown in Table I. For VR creation, the RWS interface was better than SNMPv3, followed closely by NETCONF. However, RWS and NETCONF interfaces presented equivalent average response times to remove VRs. In retrieval operations, the average response time of the NETCONF interface was very close to SNMPv2c, but in this case, the RWS interface resulted

in the highest times.

TABLE I. AVERAGE RESPONSE TIME OF MANAGEMENT INTERFACES

Interfaces	VR Creation [s]	VR Retrieval [s]	VR Removal [s]
SNMPv2c	1.02	0.03	3.41
SNMPv3	1.12	0.08	3.53
NETCONF	1.08	0.04	3.48
RWS	1.06	0.16	3.48

With regard to resource consumption, RWS achieved the highest CPU time and memory consumption. Analyzing maximum CPU time for configuration operations, there is a similarity between SNMPv2c and NETCONF, as shown in Table II. For retrieval operations, NETCONF was the interface that consumed less CPU. Regarding memory consumption, the SNMPv2c, SNMPv3, and NETCONF interfaces had much lower memory consumption than the RWS interface. Table III shows that, with respect to average memory consumption, the SNMPv2c interface was slightly better than SNMPv3,

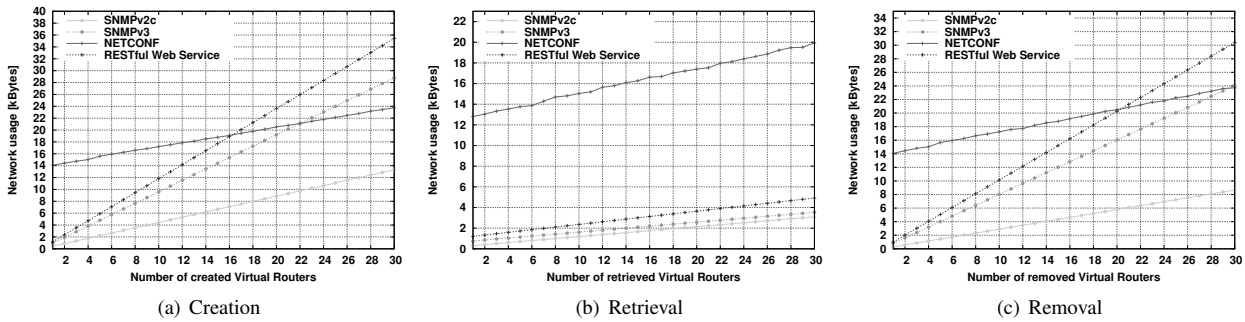


Fig. 10. Network usage of the management interfaces

followed by NETCONF.

TABLE II. MAXIMUM CPU TIME OF MANAGEMENT INTERFACES

Interfaces	VR Creation [ms]	VR Retrieval [ms]	VR Removal [ms]
SNMPv2c	7.7	9.3	8.3
SNMPv3	8.7	7.3	9.3
NETCONF	8.0	2.0	8.3
RWS	13.0	14.0	11.0

TABLE III. AVERAGE MEMORY CONSUMPTION OF MANAGEMENT INTERFACES

Interfaces	VR Creation [MB]	VR Retrieval [MB]	VR Removal [MB]
SNMPv2c	3.2	3.2	3.2
SNMPv3	3.5	3.5	3.5
NETCONF	3.9	3.8	3.8
RWS	73.1	73.2	73.1

The experiments related to network usage showed that the SNMPv2c management interface generated less traffic compared to the others. For retrieval operations, NETCONF resulted in a higher network usage than the other interfaces, which had similar results. For configuration operations, NETCONF started with a high network usage, but outperformed both RWS and SNMPv3 as the number of managed VRs increased. Table IV classifies the management interfaces in a descending order according to the achieved results for each evaluated metric.

TABLE IV. CLASSIFICATION OF MANAGEMENT INTERFACES ACCORDING TO THE EVALUATED METRICS

Metrics	VR Creation	VR Retrieval	VR Removal
Response time	SNMPv2c RWS NETCONF SNMPv3	SNMPv2c NETCONF SNMPv3 RWS	SNMPv2c RWS≡NETCONF SNMPv3
CPU time	SNMPv2c NETCONF SNMPv3 RWS	NETCONF SNMPv3 SNMPv2 RWS	SNMPv2c≡NETCONF SNMPv3 RWS
Memory consumption	SNMPv2c SNMPv3 NETCONF RWS	SNMPv2c SNMPv3 NETCONF RWS	SNMPv2c SNMPv3 NETCONF RWS
Network usage	SNMPv2c NETCONF SNMPv3 RWS (#VR > 22)	SNMPv2c SNMPv3 RWS NETCONF	SNMPv2c NETCONF SNMPv3 RWS (#VR > 30)

V. CONCLUSION

In this article, we investigated management interfaces based on SNMPv2c, SNMPv3, NETCONF, and RWS for managing physical routers supporting virtualization. We compared the interfaces by evaluating the performance of each one in terms of response time, CPU time, memory consumption, and network usage for three basic VR management operations: creation, retrieval, and removal.

Comparing the management interfaces, it is possible to observe that the SNMPv2c presented the best results for most evaluated metrics. The second best one was the NETCONF interface, which achieved results quite close to the SNMPv2c interface. Although NETCONF has presented higher network usage for few managed VRs, it demonstrated to be more scalable than SNMPv3 and RWS solutions. In addition, unlike SNMPv2c, NETCONF provides security since it uses SSH as the transport mechanism. Therefore, the SNMPv2c interface is appropriate for managing small NVEs without strict security requirements, while NETCONF appears as a promising alternative to compose a management interface to be deployed in more realistic scenarios, *i.e.*, large-scale NVEs with many VRs, where security and scalability are important concerns.

As future work, we intend to evaluate a management interface based on SOAP Web services. We also plan to investigate NETCONF combined with compression support (*zlib*) and RWS running over HTTPS. Furthermore, we look for an alternative implementation for the RWS interface, because using Java/Tomcat resulted in high resource consumption.

REFERENCES

- [1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005. [Online]. Available: <http://dx.doi.org/10.1109/MC.2005.136>
- [2] F. Daitx, R. Esteves, and L. Granville, "On the use of SNMP as a management interface for virtual networks," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 177–184. [Online]. Available: <http://dx.doi.org/10.1109/INM.2011.5990689>
- [3] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine*, vol. 47, no. 7, pp. 20–26, Jul. 2009. [Online]. Available: <http://dx.doi.org/10.1109/MCOM.2009.5183468>
- [4] —, "A survey of network virtualization," *Computer Network*, vol. 54, no. 5, pp. 862–876, Apr. 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2009.10.017>

- [5] E. Stelzer, S. Hancock, B. Schliesser, and J. Laria, "Virtual Router Management Information Base Using SMIPv2," <http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04>, Jul. 2005, internet draft. [Online]. Available: <http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04>
- [6] K. Tata, "Definitions of Managed Objects for Virtual Router Redundancy Protocol Version 3 (VRRPv3)," RFC 6527 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6527.txt>
- [7] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," RFC 3410 (Informational), Internet Engineering Task Force, Dec. 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3410.txt>
- [8] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6241.txt>
- [9] W. Stallings, "Snmv3: A security enhancement for snmp," *Communications Surveys & Tutorials, IEEE*, vol. 1, no. 1, pp. 2–17, 1998.
- [10] B. Hedstrom, A. Watwe, and S. Sakthidharan, "Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions," Ph.D. dissertation, PhD thesis, MasterâAZs thesis, University of Colorado, 2011. [Online]. Available: <http://morse.colorado.edu/~tlen5710/11s/>
- [11] A. Pras, T. Drevers, R. van de Meent, and D. Quartel, "Comparing the performance of SNMP and Web services-based management," *Network and Service Management, IEEE Transactions on*, vol. 1, no. 2, pp. 72–82, Dec 2004. [Online]. Available: <http://dx.doi.org/10.1109/TNSM.2004.4798292>
- [12] H. Asai, M. MacFaden, J. Schoenwaelder, Y. Sekiya, K. Shima, T. Tsou, C. Zhou, and H. Esaki, "Management Information Base for Virtual Machines Controlled by a Hypervisor," Jul. 2014, draft-ietf-opsawg-vmm-mib-01. [Online]. Available: <http://tools.ietf.org/html/draft-ietf-opsawg-vmm-mib-01>
- [13] NET-SNMP, accessed: Aug. 2014. [Online]. Available: <http://www.net-snmp.org/>
- [14] OpenYUMA, accessed: Aug. 2014. [Online]. Available: <https://github.com/OpenClovis/OpenYuma>
- [15] Jersey, "Restful web services in java," accessed: Aug. 2014. [Online]. Available: <https://jersey.java.net/>
- [16] XenServer, "Open Source Virtualization," accessed: Aug. 2014. [Online]. Available: <http://www.xenserver.org/>
- [17] Brocade, "SDN+NFV Develop and Enhance," accessed: Aug. 2014. [Online]. Available: <http://community.brocade.com/t5/SDN-NFV/ct-p/SdnNfv>
- [18] N. Lippis III, "Network Virtualization: The New Building Blocks of Network Design," Oct. 2007, white Paper. [Online]. Available: http://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/network-fabric/net_implementation_white_paper0900aecd80707cb6.pdf
- [19] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the Management of Virtual Networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 2–10, Jul. 2013. [Online]. Available: <http://dx.doi.org/10.1109/MCOM.2013.6553682>
- [20] Internet Engineering Task Force (IETF), "Layer 3 Virtual Private Networks (l3vpn)," accessed: Aug. 2014. [Online]. Available: <http://datatracker.ietf.org/wg/l3vpn/charter/>
- [21] R. Patricio, B. Batista, J. Junior, and A. Patel, "Proposal for a NETCONF Interface for Virtual Networks in Open vSwitch Environments," in *AICT 2014, The Tenth Advanced International Conference on Telecommunications*, 2014, pp. 57–62.
- [22] Open vSwitch, "An open virtual switch." [Online]. Available: <http://openvswitch.org/>
- [23] O. M. C. Rendon, C. R. P. dos Santos, A. S. Jacobs, and L. Z. Granville, "Monitoring virtual nodes using mashups," *Computer Networks*, vol. 64, no. 0, pp. 55 – 70, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128614000498>
- [24] J. Yu, B. Benattallah, F. Casati, and F. Daniel, "Understanding mashup development," *Internet Computing, IEEE*, vol. 12, no. 5, pp. 44–52, 2008.
- [25] C. dos Santos, R. Bezerra, J. Ceron, L. Granville, and L. Rockenbach Tarouco, "On using mashups for composing network management applications," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 112–122, December 2010. [Online]. Available: <http://dx.doi.org/10.1109/MCOM.2010.5673081>
- [26] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: <http://www.ietf.org/rfc/rfc6020.txt>
- [27] J. Schoenwaelder, "Translation of Structure of Management Information Version 2 (SMIPv2) MIB Modules to YANG Modules," RFC 6643 (Proposed Standard), Internet Engineering Task Force, Jul. 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6643.txt>
- [28] P. JAXB, accessed: Aug. 2014. [Online]. Available: <https://jaxb.java.net/>
- [29] U. Blumenthal and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFC 5590. [Online]. Available: <http://www.ietf.org/rfc/rfc3414.txt>
- [30] U. Blumenthal, F. Maino, and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model," RFC 3826 (Proposed Standard), Internet Engineering Task Force, Jun. 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3826.txt>
- [31] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFCs 5343, 5590. [Online]. Available: <http://www.ietf.org/rfc/rfc3411.txt>
- [32] M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)," RFC 6242 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6242.txt>
- [33] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.
- [34] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc7230.txt>
- [35] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (INTERNET STANDARD), Internet Engineering Task Force, Jan. 2005, updated by RFCs 6874, 7320. [Online]. Available: <http://www.ietf.org/rfc/rfc3986.txt>
- [36] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc7231.txt>
- [37] Java.net, accessed: Aug. 2014. [Online]. Available: <https://jax-rs-spec.java.net/>
- [38] Apache, "Tomcat," accessed: Aug. 2014. [Online]. Available: <http://tomcat.apache.org/>
- [39] Oracle, "Java SE at a Glance," accessed: Aug. 2014. [Online]. Available: <http://www.oracle.com/technetwork/java/javase/>
- [40] cURL, "cURL groks URLs," accessed: Aug. 2014. [Online]. Available: <http://curl.haxx.se/>
- [41] PROCPS, "The /proc file system utilities," accessed: Aug. 2014. [Online]. Available: <http://procps.sourceforge.net/>
- [42] TCPDUMP, accessed: Aug. 2014. [Online]. Available: <http://www.tcpdump.org/>