UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EFICIÊNCIA E AUTO-ESCALABILIDADE NA VIRTUALIZAÇÃO DO SERVIÇO DE TRADUÇÃO DE ENDEREÇOS

EMERSON ROGÉRIO ALVES BAREA

ORIENTADOR: PROF. DR. CESAR AUGUSTO CAVALHEIRO MARCONDES

São Carlos – SP

Fevereiro/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EFICIÊNCIA E AUTO-ESCALABILIDADE NA VIRTUALIZAÇÃO DO SERVIÇO DE TRADUÇÃO DE ENDEREÇOS

EMERSON ROGÉRIO ALVES BAREA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Redes e Sistemas Distribuídos

Orientador: Prof. Dr. Cesar Augusto Cavalheiro Marcondes

São Carlos – SP

Fevereiro/2016

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar
Processamento Técnico
com os dados fornecidos pelo(a) autor(a)

Barea, Emerson Rogério Alves
B248e Eficiência e auto-escalabilidade na virtualização do serviço de tradução de endereços / Emerson Rogério Alves Barea. -- São Carlos : UFSCar, 2016.
62 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2016.

1. NFV. 2. ClickOS. 3. PI. 4. Viabilidade. 5. Escalabilidade. I. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Emerson Rogério Alves Barea, realizada em 22/02/2016:

Prof. Dr. Cesar Augusto Cavalheiro Marcondes
UFSCar

Prof. Dr. Fabio Luciano Verdi
UFSCar

Prof. Dr. Christian Rodolfo Esteve Rothenberg
UNICAMP

Dedico o resultado desse trabalho à minha família, orientador e amigos.

AGRADECIMENTOS

Agradeço primeiramente à Deus pela minha vida, saúde, e integridade física. Aos meus pais por terem sido ótimas referências em minha vida, meu orientador Prof. Dr. Cesar Marcondes por ter confiado e me dado a chance de alcançar esse momento tão importante, aos velhos e novos amigos que por vezes tiveram participações tão importantes nessa jornada e, principalmente, à minha esposa Solange e filho Pedro, que ao mesmo tempo são pilares de sustentação, combustível e pousada para meu corpo e mente. Tenho certeza que sem vocês, nada ou muito pouco seria capaz.

RESUMO

Este trabalho apresenta uma proposta de arquitetura de serviço de tradução de endereços (NAT) com escalabilidade eficiente através do uso de Virtualização de Funções de Rede (NFV) em ambientes computacionais de baixo custo. Para isso, uma função de rede virtualizada (VNF) do tipo NAT foi instanciada em sistema minimalista ClickOS e com um sistema de controle retroalimentável realizando o controle da vazão de maneira suave. Nossos resultados indicam que a arquitetura proposta, implementada e testada atende vários requisitos considerados importantes em NFV, como alta eficiencia, obtendo ganhos de 900% na vazão em alta escala contrastado com NAT do Linux. Além disso, resultados do sistema de controle do tipo Proporcional Integral (PI) demonstram 85% de acurácia na vazão prevista em apenas 3 amostras.

Palavras-chave: NFV, ClickOS, PI, Viabilidade, Escalabilidade

ABSTRACT

This work presents a novel architecture for the address translation service (NAT) with efficient scalability through the use of Network Functions Virtualization (NFV) in low cost computing environments. To this end, a virtualized network function (VNF) of NAT is instantiated in a minimal OS (ClickOS) and uses a feedback control system to smooth the rate control. Our results indicate that the proposed architecture meets several relevant NFV requirements, including high efficiency, obtaining up to 900% higher throughput gains compared to Linux NAT. In addition, the Proportional Integral control system (PI) yields 85% accuracy in estimating the exact rate using only 3 samples.

Keywords: NFV, ClickOS, PI, Viability, Scalability

LISTA DE FIGURAS

2.1 Mudança do modelo em camadas introduzido pelo paradigma NFV. Fonte: (BRISCOE, 2013) 16

2.2 Arquitetura de alto nível para NFV. Fonte: (BRISCOE, 2013) 17

2.3 Composição de NFs por orquestração em hypervisors (BRISCOE, 2013). . . . 19

2.4 Diferentes opções disponíveis para implementar aplicações virtualizadas. . . . 20

2.5 Exemplo de um serviço de rede fim-a-fim com VNFs e VNF-FGs aninhados. . 21

2.6 Comparação entre cadeias de serviços atual e de NFV. 22

2.7 Eficiência de chip vs. complexidade de caso de uso (PONGRáCZ et al., 2013). 25

2.8 Visão lógica de gerenciamento (CSáSZáR et al., 2013). 26

2.9 Relação entre SDN e NFV (NFV White Paper, 2012). 30

3.1 Arquitetura para suporte à escalabilidade eficiente de recursos NFV. 34

4.1 Arquitetura ClickOS (MARTINS et al., 2013). 38

4.2 Rede Básica e Otimizada ClickOS no Xen (MARTINS et al., 2013). 39

4.3 ClickOS: Tempos de criação, inicialização, instalação de middlebox e utilização de banda acumulada (MARTINS et al., 2013). 40

4.4 Saturação de vazão em instância Linux (a) e ClickOS (b). Taxa de transmissão em múltiplos de 20Kp/s; 50.000 endereços emissores; VNF suportando 1 e 1000 endereços de mascaramento; ***hyperthreading*** habilitado. 43

4.5 Vazão da VNF versus ***cap*** ajustado em instância Linux. Tráfego gerado iniciando em 20Kp/s, sendo acrescido em múltiplos de 20Kp/s. O ***cap*** iniciou em 5%, sendo acrescido em múltiplos de 5%. 44

4.6 Vazão da VNF versus ***cap*** ajustado em instância ClickOS. Tráfego gerado iniciando em 20Kp/s, sendo acrescido em múltiplos de 20Kp/s. O ***cap*** iniciou em 5%, sendo acrescido em múltiplos de 5%. 45

5.1 Diagramas de Bloco de Sistemas de Controle de Malha Aberta e Malha Fechada. 48

5.2 Diagrama de Blocos do Sistema de Controle de Malha Fechada. 53

5.3 Componentes do sistema de controle para escalabilidade automática de recursos NFV. R(z) é a vazão esperada. Y(z) a vazão resultante da VNF. E(z) o erro observado entre R(z) e Y(z). K(z) é a função de transferência do ***Controlador PI***, U(z) o sinal de controle gerado por K(z). G(z) a função de transferência do ***Processo Xen***. F(z) a função de transferência de todo sistema de controle. . . . 53

5.4 Resíduos do modelo. 54

LISTA DE TABELAS

4.1 Configuração de hardware das máquinas utilizadas nos experimentos. 41

4.2 Hardware virtual das instâncias de VNF NAT. 42

5.1 Ações PID e seus Pontos de Atuação e Capacidade de Equilíbio com Zero Erro. 51

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO E MOTIVAÇÃO **11**

1.1 Contexto 11

1.2 Motivação e Objetivos 12

1.3 Organização do trabalho 12

CAPÍTULO 2 – VIRTUALIZAÇÃO DE FUNÇÕES DE REDE **14**

2.1 Principais objetivos e Metas de NFV 14

2.2 Modelo em camadas 15

2.3 Arquitetura de Referência 16

2.3.1 Blocos Funcionais 17

2.4 Composição de Serviços de Redes 20

2.5 Requisitos e Desafios de NFV 21

2.5.1 Especificações de VFNGs (cadeias de serviços) 22

2.5.2 Desempenho e escalabilidade 24

2.5.3 Gerenciamento, Orquestração e Automação de NFs 25

2.5.4 Portabilidade e Interoperabilidade com Plataformas Legadas 27

2.5.5 Estabilidade da rede: Segurança e Resiliência 28

2.5.6 NFV e SDN 30

CAPÍTULO 3 – ARQUITETURA PROPOSTA **32**

3.1 Requisitos para Escalabilidade Eficiente de Recursos NAT em NFV 32

3.1.1 Requisitos gerais do NFV 32
3.1.2 Requisitos específicos da VNF NAT 33
3.1.3 Requisitos de controle da escalabilidade da VNF 33
3.2 Arquitetura 34
3.3 Implementação e Análise de Desempenho 35

CAPÍTULO 4 – SISTEMA MINIMALISTA COMO BASE DA VNF 36

4.1 ClickOS 36
4.2 Avaliação do Sistema Minimalista como Base da VNF 40
4.2.1 Ambiente Experimental 40
4.2.2 Definições e Configurações Preliminares 41
4.2.3 Saturação da vazão 42
4.2.4 Vazão versus ***cap*** ajustado 44

CAPÍTULO 5 – SISTEMA DE CONTROLE DE VAZÃO PARA VNF 46

5.1 Sistemas de Controle 46
5.1.1 Conceitos e Definições 46
5.1.2 Algoritmo Proportional Integral Derivativo (PID) 49
5.1.3 PID em Sistemas de Controle Computacionais 51
5.2 Sistema de Controle para Escalabilidade Eficiente de VNF 52
5.2.1 Diagrama de Blocos 52
5.2.2 Processo Xen 53
5.2.3 Controlador PI 55

CAPÍTULO 6 – TRABALHOS RELACIONADOS 56

CAPÍTULO 7 – CONCLUSÃO E TRABALHOS FUTUROS 58

REFERÊNCIAS 59

Capítulo 1

INTRODUÇÃO E MOTIVAÇÃO

Este capítulo apresenta a introdução ao tema tratado neste trabalho. A Seção 1.1 contextualiza o tema principal; a Seção 1.2 trata das motivações e objetivos; e a Seção 1.3 descreve a estrutura deste documento.

1.1 Contexto

Historicamente, o provimento de serviços de redes de computadores e telecomunicação está atrelado à grandes fabricantes, reconhecidos mundialmente, os quais são responsáveis pelo fornecimento de *software* e *hardware* proprietários destinados as operadoras. Tal dependência acontece, principalmente, devido ao fato das tecnologias oferecidas atenderem as necessidades críticas de desempenho, estabilidade e disponibilidade requeridas nesse segmento de mercado.

Essa situação de *lock-in* traz dificuldade, risco e algum prejuízo à inovação aos consumidores dessas tecnologias. Haja visto que, por vezes, há necessidade de instalação de diversos equipamentos (ex: *middleboxes*) proprietários e inflexíveis. Esses equipamentos necessitam de gerenciamento altamente especializado e licenciamento específicos. Também são comuns a falta de integração entre as tecnologias de diferentes fabricantes, lentidão no desenvolvimento de soluções específicas às necessidades do cliente final, dificuldades na contratação de mão-de-obra qualificada, alto custo de implantação e elevado consumo energético.

Recentemente, um grupo de operadoras de telecomunicação e fabricantes de equipamentos uniram-se para desenvolver um novo conceito de provimento de serviços de rede e telecomunicação: Network Function Virtualisation (NFV) (ROSA et al., 2014). Esse grupo tem objetivo de tentar solucionar as dificuldades apresentadas. Em sua essência, NFV faz uso do processo de consolidação de tecnologias, que acontece há anos em centros de dados, somente recentemente

adaptado para a área de redes. Isso corresponde ao oferecimento das funções de rede através de instâncias virtualizadas em servidores de uso geral. As vantagens do NFV vão de encontro às limitações das tecnologias proprietárias convencionais de rede e aceleram a inovação. Por outro lado, também trazem novos desafios como: interoperabilidade, alto desempenho e a portabilidade das funções de rede virtualizadas (VNF - Virtual Network Function).

1.2 Motivação e Objetivos

Outro aspecto importante e possível com o uso de NFV em telecomunicações é a escalabilidade do ambiente, dado que os componentes são virtuais. Esses componentes podem ser orquestrados de modo a garantir a escalabilidade automática do ambiente de forma a atender às necessidades de recursos impostas pela VNF, assim como acontece com serviços web na computação em nuvem, que são auto-escaláveis.

Baseado nesse cenário, este trabalho apresenta uma proposta de auto-provisionamento otimizado de VNF para o serviço de tradução de endereços de rede (NAT). Entre os destaques e aspectos inovadores, esta proposta (*i*) faz uso de equipamentos de baixo custo, (*ii*) maximiza a eficiência utilizando tecnologia de sistema operacional minimalista (ClickOS), e (*iii*) fornece auto-escalabilidade como um sistema otimizado de controle de vazão retroalimentável.

A nossa escolha do serviço NAT é justificada, em função de ser um serviço bastante comum em operadoras, carentes de endereçamento IPv4. Dessa forma, nossa arquitetura foi modelada para atender NAT, contudo, também esta estruturada de forma genérica o suficiente para atender outras VNFs. Nossos resultados indicam que a arquitetura atende vários requisitos considerados importantes em NFV como alta eficiencia, através do uso sistema minimalista ClickOS processando NAT, com ganhos de até 900% na vazão. Os resultados do sistema de controle retroalimentável do tipo Proporcional Integral (PI) demonstram 85% de acurácia na vazão prevista em apenas 3 amostras, facilitando um controle refinado e a suavização do tráfego.

1.3 Organização do trabalho

O restante desse trabalho está organizado da seguinte maneira: o Capítulo 2 detalha o conceito de NFV. No Capítulo 3 são apresentados a motivação e a arquitetura genérica do sistema de auto-provisionamento. O Capítulo 4 detalha e discute os resultados da análise de desempenho do sistema minimalista utilizado como base da VNF. O Capítulo 5 conceitua teoria de controle, apresenta o algoritmo PID e discute os resultados do emprego de sua variante PI no controle

eficiente da escalabilidade de VFN. O Capítulo 6 apresenta e discute os trabalhos relacionados ao tema. O Capítulo 7 conclui este trabalho e apresenta sugestões de trabalhos futuros.

Capítulo 2

VIRTUALIZAÇÃO DE FUNÇÕES DE REDE

Neste capítulo são apresentados os principais conceitos relativos a NFV. A Seção 2.1 apresenta os objetivos e metas de NFV. Em seguida é apresentado o modelo de camadas de NFV na Seção 2.2, seguido do detalhamento de sua arquitetura de referência e blocos funcionais na Seção 2.3. A Seção 2.4 complementa o entendimento da arquitetura de NFV detalhando a composição dos serviços de rede. Por fim, a Seção 2.5 apresenta e detalha os desafios inerentes a NFV.

2.1 Principais objetivos e Metas de NFV

O NFV conforme sendo discutido pelo Grupo de Trabalho da ETSI [1] possui os seguintes objetivos principais:

Proporcionar eficiência nos gastos de capital (CAPEX) comparando-se a implementações com *hardware* **dedicado.** Este objetivo é alcançado utilizando *hardware* comercial de uso geral, também chamado *hardware* comoditizado (ex: servidores e *storage*), para prover funções de dispositivos de rede através de técnicas de virtualização. O *hardware* comoditizado tem a vantagem de ser adquirido com ganhos de economia de escala, e o processador genérico pode ser flexível à diferentes tipos de carga de trabalho. Além disso, o compartilhamento de *hardware* para realização de diferentes funções, resulta na redução dos tipos de dispositivos proprietários utilizados na rede, e podem auxiliar na redução de CAPEX. Na prática, espera-se que isso ocorra a médio e longo prazo, visto que inicialmente pode ser necessário investimentos para melhorar a capacidade de desempenho dos equipamentos de *hardware* genérico já existentes nas empresas.

[1] http://www.etsi.org/technologies-clusters/technologies/nfv

Flexibilidade na designação espacial das funções de rede para *hardware* **de propósito geral.** Isto permite obter escalabilidade, já que desacopla as funcionalidades de *hardware* e locais específicos. Também é possível que as funções (*software*) virtualizadas sejam posicionadas de forma flexível nos locais mais apropriados. Esses locais são chamados de NFVI-PoPs. E as funcionalidades migram para esses locais, dependendo de condições externas como demanda, falhas, horário do dia, política de compartilhamento de recursos, ciclo de liberação (Ex.: versões alfa, beta, produção). Os NFVI-PoPs podem incluir locais como *sites* de clientes, pontos de troca de tráfego (PTTs), centrais de operadoras, data centers, entre outros.

Rápida implementação de novos serviços sem necessidade de alterações em plataformas de *hardware*. Somente pelo desenvolvimento e implantação de novos *software*s ou versões de *software*.

Melhora de eficiência nos processos operacionais através da automatização destes processos, resultando em uma substancial redução em custos operacionais (OPEX).

Redução na utilização de energia elétrica contribuindo com as iniciativas *green networks*, já que permite a migração de cargas e desligamento de *hardware* não utilizado, e os próprios ganhos de energia com a consolidação.

Padronização e abertura de interfaces entre funções virtualizadas e entidades de gerenciamento da infraestrutura, de forma que os diferentes elementos possam ser oferecidos por diferentes fornecedores de forma desacoplada.

2.2 Modelo em camadas

Para atender os objetivos acima, a implementação das funcionalidades de rede em ambientes de computação virtualizada apresenta algumas mudanças no modelo de camadas OSI. O modelo OSI, até então, era considerado referência na descrição de arquiteturas de redes de computadores. A Figura 2.1 ilustra a dependência típica de uma aplicação de rede, na pilha de sistemas e das camadas inferiores. O típico são as funções de rede estarem na Camada de Rede OSI. Entretanto, no modelo NFV (Fig.2.1 à direita) as funções de rede não estão mais diretamente dependentes da infraestrutura física de comunicação. Ou seja, elas podem ser implementadas no domínio puramente de *software*, na camada de Aplicação OSI de alto nível. E portanto, dependentes das camadas do sistema operacional e de virtualização presentes na infraestrutura computacional. O *trade-off* entre desempenho e flexibilidade (assim como as implicações de segurança) do modelo NFV é um dos aspectos principais introduzidos nesse novo paradigma de redes.

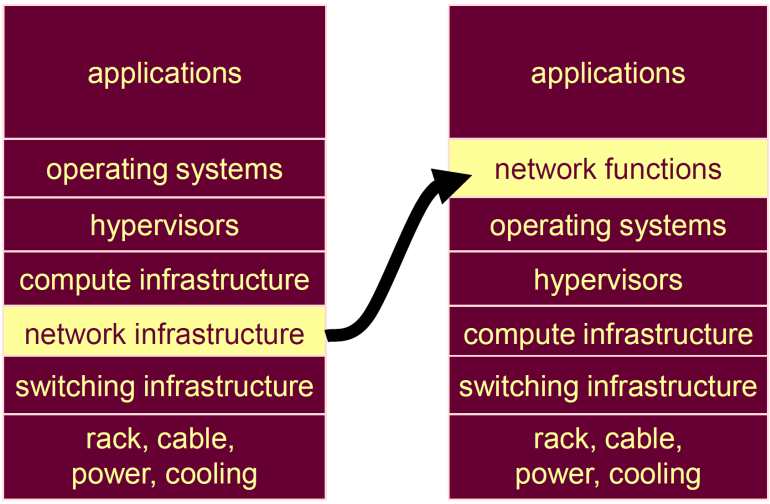


Figura 2.1: Mudança do modelo em camadas introduzido pelo paradigma NFV. Fonte: (BRISCOE, 2013)

2.3 Arquitetura de Referência

Buscando viabilizar o desenvolvimento do NFV, o grupo de trabalho do ETSI (2013) apresentou uma arquitetura de referência responsável por delinear aspectos de virtualização que garantam o funciomento e evolução do NFV. Os principais aspectos abordados na elaboração da arquitetura foram:

- Correto funcionamento das VNFs mesmo sobre plataformas de *software* e *hardware* (ex: hypervisores, servidores, *storages* etc) variados;
- Delinear funções de rede como blocos funcionais, permitindo a construção de grafos de encadeamento de VNFs (VNFs FG);
- Existência de interfaces de gerenciamento e orquestração compatíveis com estruturas legadas;
- Suporte a diferentes níveis de SLA;
- Ausência de novos problemas de segurança, desempenho e compatibilidade.

A Figura 2.2, ilustra a arquitetura de alto nível para virtualização de funções de redes. Essa arquitetura é dividida em três blocos funcionais principais, detalhados posteriormente:

- Funções de rede virtualizadas (VNFs)
- Infraestrutura NFV (NFVI)

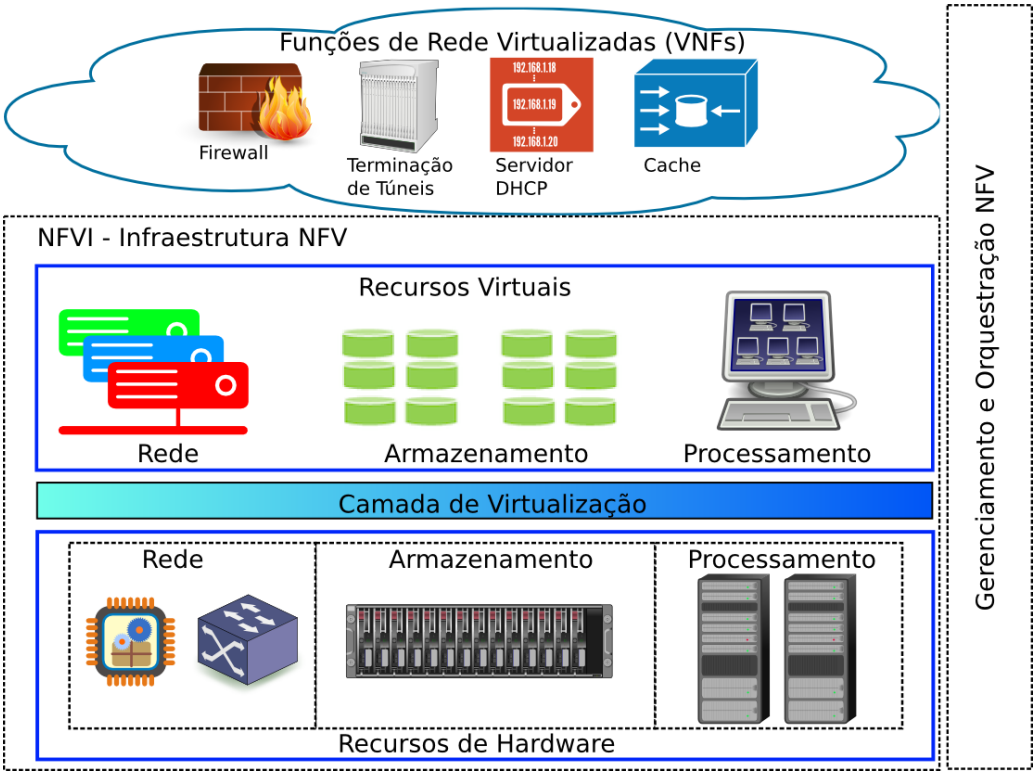


Figura 2.2: Arquitetura de alto nível para NFV. Fonte: (BRISCOE, 2013)

- Gerenciamento e orquestração NFV.

2.3.1 Blocos Funcionais

Funções de Rede Virtualizadas (VNF): Exemplos de funções de rede incluem as desempenhadas por roteadores, *firewalls*, *gateways* residenciais, elementos da arquitetura 3GPP (*3rd Generation Partnership Project*) como o MME (*Mobile Management Entity*), PGW (*Packet Data Network Gateway*), servidores de autenticação, servidores DHCP (*Dynamic Host Configuration Protocol*), entre outros. Uma função de rede pode ser decomposta em diferentes componentes externos, os quais podem sem implementados em diferentes máquinas virtuais ((*Virtual Machine*). No entanto, o comportamento funcional de uma função de rede não depende se tal função é virtualizada em múltiplas VM, em uma única VM, ou não é virtualizada.

Infraestrutura para NFV (NFVI): NFVI é a composição dos recursos de *hardware* e *software* necessários para a implementação, execução, e gerenciamento das VNFs. A infraestrutura para provimento de NFVI pode ser distribuída em diferentes NFVI-PoPs, de forma que a rede que provê conectividade entre os NFVI-PoPs também faz parte da NFVI. No entanto, a camada de virtualização para os recursos de *hardware* visto pelas VNFs permite que a NFVI seja uma entidade única. Adicionalmente, na NFVI, os **recursos de** *hardware* são responsáveis por pro-

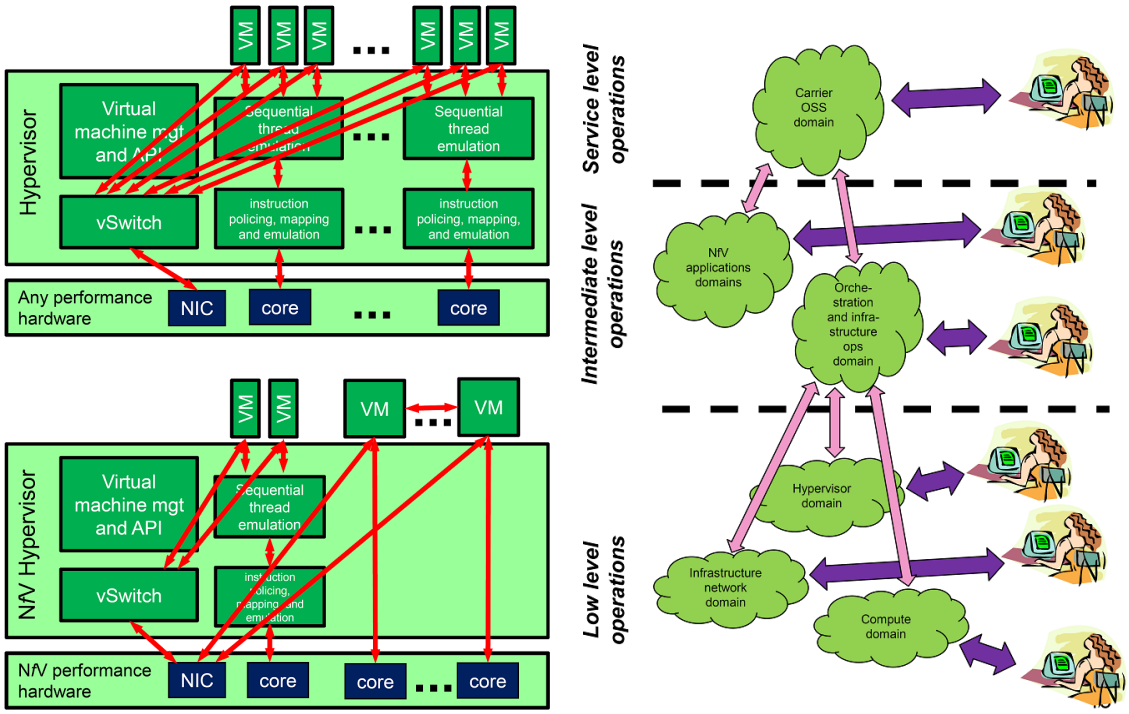
ver infraestrutura de processamento, armazenamento e conectividade para as VNFs através da camada de virtualização. Desse modo, é fundamental que os recursos de *hardware* sejam de propósito geral, referidos como COTS (*Commercial off-the-shelf*), os *hardware*s comoditizados. Para interconectar esses recursos, há dois tipos de redes: as redes internas de *datacenter* dos NFVI-PoP e as redes de transporte que os interconectam.

A **camada de virtualização** também é importante em NFVI, pois ela abstrai os recursos de *hardware*, desacoplando o *software* das VNFs do *hardware* especializado, garantindo independência do *hardware* utilizado. Isto é feito através do particionamento lógico dos recursos físicos (*slicing*) entre as VNFs, provendo os recursos lógicos para as VNFs através da camada de abstração de *hardware*. Tipicamente, o desacoplamento dos recursos de computação e armazenamento do *software* é realizado através de hipervisores, como KVM (*Kernel-based Virtual Machine*), VMWare, Xen e outros, permitindo a execução das VMs em *hardware* de propósito geral. No entanto, é importante notar que uma VNF poderia ser executada em uma ou mais VMs. Desse modo, a arquitetura de referência para NFV não define uma solução específica para a camada de virtualização. Em outras palavras, ainda está em aberto a definição de um padrão mais adequando para a abstração do *hardware*. De qualquer modo, o uso de hipervisores é uma das soluções atuais para essa camada.

No domínio da conectividade dessa camada, várias técnicas têm sido utilizadas permitindo a virtualização de recursos de rede de forma a prover conectividade entre as VMs de uma mesma VNF. Para essa função de interconexão inter-VNFs, normalmente são usados recursos de tunelamento, como VxLANs (*Virtual Extensible Local Area Network*), NVGRE (*Network Virtualisation using Generic Routing Encapsulation*) e OpenFlow.

A recente implementação de funções avançadas de rede dentro dos hipervisores também vem colaborando com a evolução do NFV. Essas funções (ie., *bypass*) permitem o acesso de funções de processamento e rede diretamente do hipervisor para as VMs, Figura 2.3a. Outra optimização possível é a permissão de comunicação direta entre VMs, sem intervenção do hipervisor, o que corresponde a um aumento considerável no desempenho de rede para as VMs. Tais avanços dos hipervisores, adequados para o modelo NFV, são atualmente alvos de recentes pesquisas (ex: ClickOS, VALE, etc.) (TRILOGY2, 2014).

Os esforços em novos projetos de hipervisores incluem também a utilização de técnicas de virtualização leve (*Lightweight Virtualisation*). O modelo tradicional de VMs (virtualização completa ou paravirtualização) implica um alto custo em termos de consumo de recursos (memória, CPU, disco) para cada máquina virtual. Tal abordagem limita o número máximo de VMs que podem ser executadas em paralelo em cada servidor (na ordem de dezenas) assim como introduz



(a) Domínio de hipervisor **(b) Orquestração de NFs a nível de hipervisor**

Figura 2.3: Composição de NFs por orquestração em hypervisors (BRISCOE, 2013).

restrições em relação ao tempo máximo para inicialização (*bootup*) das VMs.

A figura 2.4 apresenta as diferentes hierarquias de virtualização que oferecem múltiplos ambientes de execução possíveis (servidor, VM, Linux contêineres, processo, thread). Cada ambiente virtualizado traz diferentes níveis de desempenho e isolamento, incluindo aspectos de segurança. Avanços recentes em tecnologias de virtualização, como por exemplo o ClickOS (KOHLER et al., 2000), tem reduzido o *overhead* de consumo de recursos de cada instância virtual, abrindo novas oportunidades para a escalabilidade das soluções de NFV baseadas num mapeamento otimizado entre a aplicação que implementa a função de rede é a VM onde é executada.

Gerenciamento e Orquestração: Os gerenciadores da infraestrutura virtualizada controlam a interação da VNF com os recursos físicos sob sua autoridade (ex: alocação, desalocação e inventário), além de operações como visibilidade da infraestrutura, coleta de informações para gerência de falhas e desempenho. Por outro lado, os gerenciadores das VNFs são responsáveis pelo gerenciamento do ciclo de vida das VNFs, incluindo operações como instanciação, atualização e finalização.

A constituição de aplicações de NFV sobre o modelo de hipervisor permite que *softwares* de orquestração (ex: vSphere, OpenStack, CloudStack) selecionem, configurem e inicializem VMs

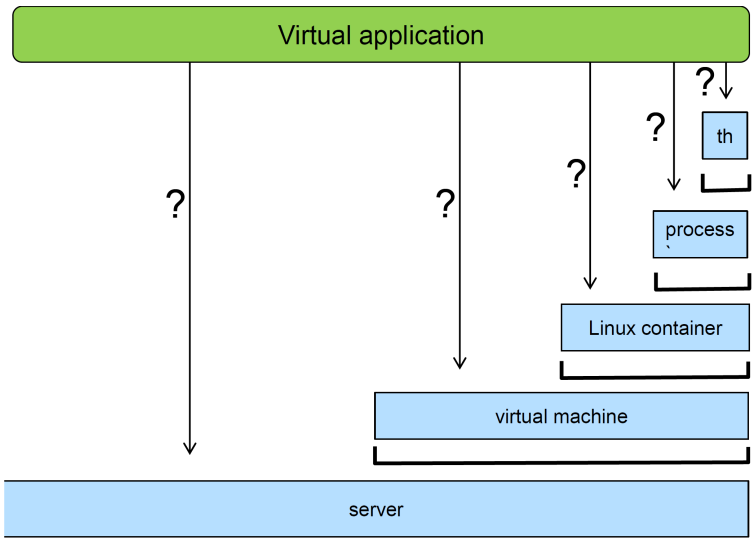


Figura 2.4: Diferentes opções disponíveis para implementar aplicações virtualizadas.

e hosts de acordo com operações de serviço de alto nível (vide Figura 2.3b) para especificação de perfis de aplicações com configurações específicas segundo sua localização e função de rede. Dessa maneira as tarefas de orquestração e gerenciamento se tornam flexíveis a ponto de constituírem cadeias de serviços facilmente programáveis em elementos nas bordas da rede.

Outras entidades envolvidas no Gerenciamento de NFV, incluem o **EMS** (*Element Management System*) que provê funções típicas de gerenciamento para uma ou mais VNFs, e o **Orquestrador**, responsável pelo gerenciamento dos serviços, orquestrando recursos de infraestrutura e de *software* para as VNFs. Complementando, há também a **Base de Configuração**, que inclui informações de configuração dos serviços das VNFs e infraestrutura, e *templates* para a implementação de VNFs, grafos de encaminhamento das VNFs e informações relativas aos serviços e infraestrutura.

2.4 Composição de Serviços de Redes

Para demonstrar a arquitetura de referência em execução, é preciso definir os serviços de rede (ex: Acesso Internet banda larga, VPN, serviço web) e esses serviços fim-a-fim podem ser descritos no ambiente NFV por um Grafo de Encaminhamento de Funções de Rede (*NF Forwarding Graph*), desta forma, o comportamento de um serviço de rede é a combinação do comportamento de seus blocos funcionais, que podem incluir NFs individuais, conjuntos de NFs, outros grafos de encaminhamento de NFs ou a própria infraestrutura de rede.

A Figura 2.5 mostra um exemplo de serviço de rede fim-a-fim e as diferentes camadas envolvidas no seu processo de virtualização. Os NFVI-PoPs possuem recursos de armazena-

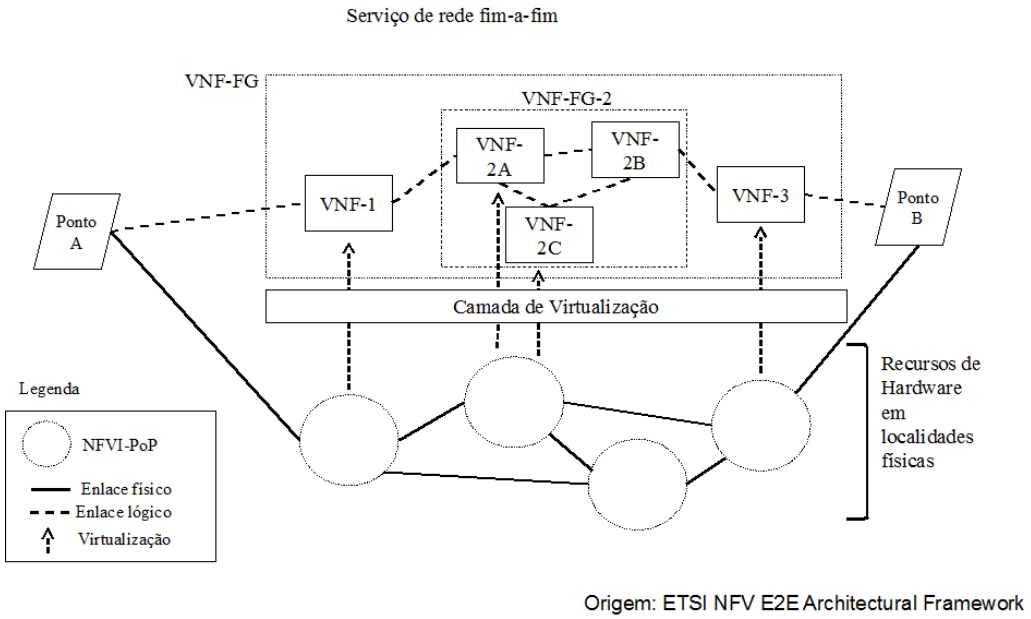


Figura 2.5: Exemplo de um serviço de rede fim-a-fim com VNFs e VNF-FGs aninhados.

mento, processamento e rede, bem como conectividade entre estes; o VNF-FG é composto por duas VNFs (VNF-1 e VNF-3) interconectadas por um VNF-FG aninhado (VNF-FG-2) que é composto por três VNFs (VNF-2A, 2B e 2C). As funções de virtualização são instanciadas sobre a Camada de Virtualização, permitindo a abstração da localização física e de quais recursos físicos são utilizados para o provimento das VNFs e do serviço.

2.5 Requisitos e Desafios de NFV

Mesmo com a definição da Arquitetura de Referência do NFV e a realização de diversos trabalhos visando seu desenvolvimento e consequente viabilidade, alguns novos questionamentos surgem e levantam requisitos ainda não contemplados. Por exemplo, a virtualização do equipamento que fica dentro das instalações do cliente (*Virtual Customer Premises Equipment* - vCPE), onde equipamentos de rede situam-se na fronteira entre usuários e provedores de serviços, traz consigo as propriedades de portabilidade e elasticidade, as quais irão suprimir custos de serviços operacionais das empresas de telecomunicação, no entanto, tais primitivas por si só já se tornam desafios em um ambiente com necessidades de resiliência e segurança (GS, 2013a).

Nesta linha, esta seção, aborda os principais desafios que NFV vem construindo com a definição de requisitos (GS, 2013c) e provas de conceito de VNFs (GS, 2014). Para abordagem do tema, os desafios são definidos em tópicos como: cadeias de serviços; trade-offs de desempenho; portabilidade, interoperabilidade de plataformas NFV e coexistência de plataformas

legadas; gerenciamento, orquestração e automação de arquiteturas e funções de rede; segurança; resiliência; e integração de SDN e NFV.

2.5.1 Especificações de VFNGs (cadeias de serviços)

O desenvolvimento de padrões de NFs bem como suas diferentes funcionalidades e interfaces de comunicação, seja com outras NFs ou recursos físicos, é uma tarefa que abrange a terminologia recomendada pelo ETSI, a qual permite um amplo espectro de características a serem propostas em provas de conceito (*Proof of Concept* - PoC)(GS, 2013b). Como representado pela terminologia, relações de VNFs constituem VNFGs que definem a composição de um ou mais serviços de rede, logo, agregar serviços a ponto de permitir a formação de cadeias é uma tarefa que exige não só a definição de características de NFs e suas respectivas interfaces, como também as ações e suas respectivas propriedades de execução sobre o tráfego a ser conduzido em um VNFG. Em termos simples, tornar uma cadeia de NFs funcional a ponto de ser utilizada em um domínio específico de rede, seria, por exemplo, agregar e orientar determinadas funcionalidades de DHCP, *firewall* e roteamento em *gateways* residenciais (ex: *Broadband Remote Access Server* - BRAS) (BIFULCO et al., 2013). Este é um requisito de NFV que agrega em si praticamente todos os desafios apresentados nesta seção, os quais já estão sendo discutidos pela comunidade acadêmica (JOHN et al., 2013) e pelo IETF (QUINN; NADEAU, 2013). A figura 2.6 representa a dualidade entre o modelo atual (em linha contínua) de cadeias de serviços e o modelo proposto por NFV (em linha pontilhada).

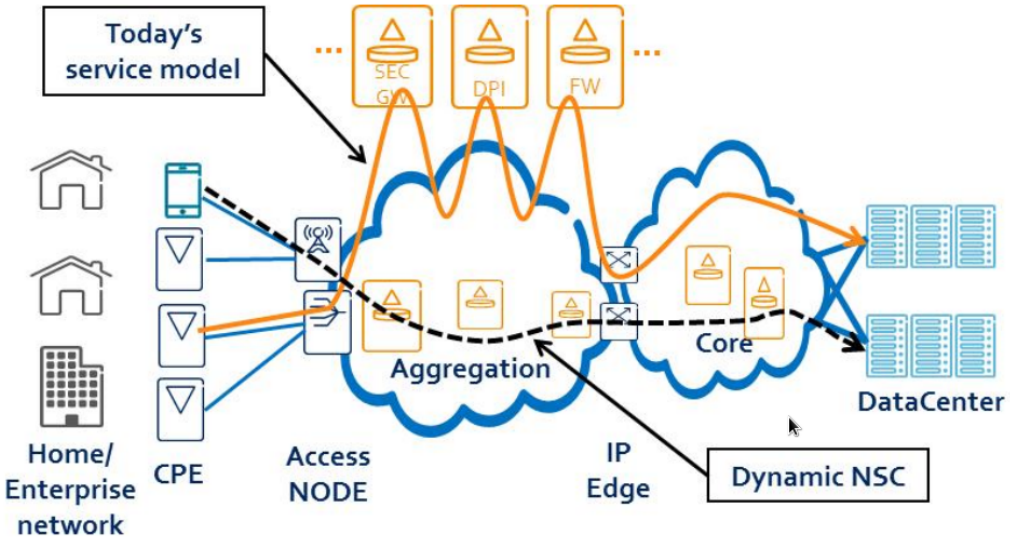


Figura 2.6: Comparação entre cadeias de serviços atual e de NFV.

Como requisitos gerais, VNFGs podem ser compostas por VNFs e PNFs, definindo um modelo NFV capaz de compor diversos serviços. Além disso, tal implementação pode ser

constituída sobre um ambiente com múltiplos provedores de serviços em NPOPs compartilhando um ambiente de NFVI com um ou mais operadores. Tomando como base este exemplo, é possivel notar os desafios que se agregam em tal ambiente, onde as relações entre VNFs e PNFs constituem particularidades que podem repercutir em outros desafios, como elasticidade e desempenho. Tais particularidades, que as interfaces entre NFs podem apresentar, estão suscetíveis ao ambiente em que elas são implementadas, como equipamentos BRAS e vCPE.

Além dos fatores de interfaces entre NFs, a própria constituição de cadeias de serviços pode trazer desafios na construção de VNFGs. Isto envolve a repartição e montagem de uma função de rede por diversas subdivisões de tarefas, como por exemplo, a tarefa de inspeção profunda de pacotes (*Deep Packet Inspection* - DPI), subdividida em um componente complexo de classificação de tráfego, um componente de cache de fluxos que mantém registro do tráfego já identificado, e outro componente que aplica as políticas pré configuradas. Nesse sentido, tais desafios, de forma semelhante a terceirização de funções de rede por subcomponentes de rede, podem ser definidos por três critérios: semânticas de processamento, garantias de desempenho e verificações de custos (FAYAZBAKHSH; REITER; SEKAR, 2013).

Semânticas de processamento dizem respeito às operações que NFs podem realizar em fluxos de tráfego, tornando estes suscetíveis a falhas de processamento por outras NFs. Por exemplo, em uma cadeia de NFs (NF1, NF2, ...), NF1 sendo um balanceador de carga pode realizar modificações nos pacotes com a consequência de que NF2, sendo por exemplo um sistema de prevenção contra intrusão (*Intrusion Prevention System* - IPS), descarte-os. Nesse caso, um provedor de serviço, utilizando esta cadeia de NFs, não terá informações se o descarte dos pacotes foi causado pela rede (ex: filas cheias) ou pela constituição errada de NFs.

No caso de **garantias de desempenho**, um provedor de serviço necessita saber que suas aplicações não sofram perda de desempenho por sobrecargas inseridas pela comunicação entre NFs de um VNFG requisitado. O desafio, neste caso, surge da ocorrência de efeitos não determinísticos causados por operações em nível de rede e pela própria rede. No caso de operações, NFs podem realizar processamento de pacotes em lote (*batch*) ou atrasá-los para eliminação de redundâncias. No segundo caso, congestionamentos por filas cheias ou falhas em computaçção de rotas podem ser os causadores comumente vistos em ambientes não virtualizados.

Sob o ponto de vista de **custo**, as cobranças realizadas sobre serviços prestados por NFs também podem ser afetadas caso VNFGs apresentem inconsistências em medições de cargas de trabalho das funcionalidades contratadas. Um cliente não é capaz de medir o consumo de recursos contratado em uma determinada NF caso esta esteja operando com anormalidades geradas por outras NFs do mesmo VNFG, nesse caso, segue como critério rígido a definição

de métodos de medição e validação de NFs bem como de suas interações com outras NFs para definição de uma cadeia de serviços.

2.5.2 Desempenho e escalabilidade

Uma das grandes premissas de NFV é o suporte de grande volume de servidores de *hardware* genérico desempenhando papéis diversos à NFVI, tornando eficiente a alocação dinâmica de recursos a diferentes VNFs. Abordar a utilização eficiente de recursos virtualizados requer que estes sejam escaláveis às dimensões dos serviços oferecidos por VNFs, assim como que estas, definidas em uma vasta heterogeneidade (ex: *firewall*, DPI, BRAS), possam utilizar aplicações do plano de dados de maneira customizada às suas necessidades (GS, 2013c).

Em escalas geográficas diferentes, partindo de grandes centros de dados a pequenas WANs, a capacidade de adequação das atuais tecnologias de virtualização, principalmente no que se diz respeito a equipamentos de rede, pode possuir diferentes peculiaridades. Por exemplo, atualmente roteadores de grandes domínios da Internet possuem grande quantidade de memória para armazenamento de suas tabelas BGP, diferenciando-se de equipamentos de borda de redes menores, onde se preza por filas e tabelas de encaminhamento pequenas devido a natureza do tráfego deste ambiente. Essa diferença de cenários, em termos de escala e requisitos de operação, é um dos fatores que tornam complexo o desenvolvimento de tecnologias de virtualização de funções de rede flexíveis e elásticas (MANZALINI; SARACCO, 2013).

Quanto aos requisitos de desempenho, é necessário considerar que uma instância de VNF deve ter suas especificações de desempenho bem definidas para operar conforme os recursos disponíveis da infraestrutura compartilhada/isolada que a hospeda, do mesmo modo, as formas de coletar informações sobre armazenamento, rede e processamento de VNFs também devem ser bem definidas e, consequentemente, realizadas em diferentes níveis de infraestrutura (ex: hipervisores, servidores, VMs). Esse procedimento garante estados consistentes de NFs e seus respectivos ambientes.

Inicialmente, há grande probabilidade que as atuais tecnologias de virtualização não apresentem comportamentos tão bons quanto *middleboxes* dedicados, mas trarão a flexibilidade necessária para fornecer elasticidade às VNFs (ex: BRAS (BIFULCO et al., 2013)).

Tecnologias habilitadoras existentes (ex: Intel DPDK, ClickOS) já atendem aos requisitos de operação esperados por diferentes escalas de usuários e NFs, sendo suporte para os demais desafios de NFV, tais como balanceamento de carga dinâmico e automatizado. É importante analisar, como apresentado na figura 2.7 (eixo x em escala contínua e eixo y em escala loga-

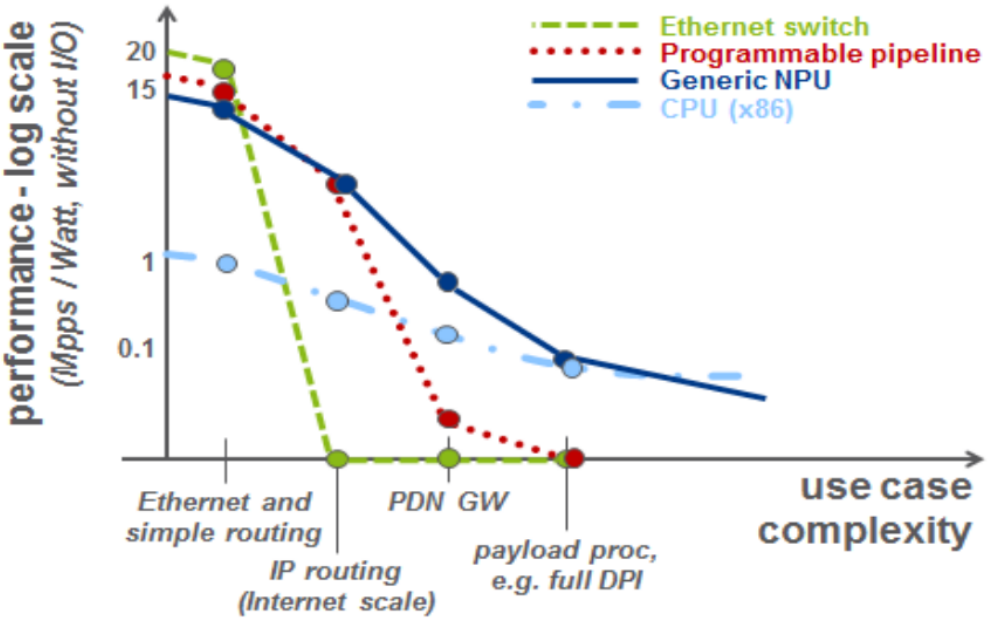


Figura 2.7: Eficiência de chip vs. complexidade de caso de uso (PONGRáCZ et al., 2013).

ritmica), a relação entre a **complexidade de uso** versus a **eficiência de chips** utilizados em ambientes habilitadores ao conceito de NFV. Na figura é possível verificar que a complexidade de uso cresce conforme o desempenho de todas as soluções decrescem; chipsets de propósito único são limitados, e não são indicados em cenários com tarefas mais complexas; e CPUs genéricas apresentam desempenho superior às NPUs em casos de uso complexos, tais como métricas baseadas em classificação de pacotes (ex: DPI).

2.5.3 Gerenciamento, Orquestração e Automação de NFs

Em um ambiente onde NFs se tornam objetos com propriedades de mobilidade, podendo ser inseridas em diversos contextos e ambientes conforme demanda ou requisitos de operação, o gerenciamento e orquestração de NFs são tarefas críticas que colaboram com a agilidade da rede, definindo seu desempenho na tradução de objetivos de alto nível em procedimentos e operações consistentemente cadenciadas, as quais podem ser até mesmo automatizadas.

Em uma NFVI, a heterogeneidade de tecnologias de hipervisores, NFs e provedores de serviços não dispõem de uma linguagem única de comunicação, nesse caso, protocolos de descoberta de serviços (*Service Discovery Protocols* - SDP) detém a tarefa crucial de serem arquitetados a operar com os seguintes parâmetros e desafios: linguagem de descrição de serviços; formato de mensagens; arquitetura de diretórios; e comportamentos de operação e comunicação em rede.

Como abordado na subseção 2.5.1 serviços de NFV podem ser arquitetados desde a constituição de VMs até a programação de cadeias de regras em switches para direcionamentos de fluxos de tráfego para NFs e PFs em servidores ou *middleboxes*. A necessidade de abstrações consistentes para a elaboração de NFs parte do pressuposto da tradução de objetivos lógicos de alto nível para funções de APIs que podem desencadear séries de comandos na infraestrutura de rede subjacente. Esta compilação, propriamente definida como orquestração, requer o uso de interfaces (ex: REST, JSON, XML) que abstraiam visões amplas da rede (ex: topologia, recursos e características de enlaces) abrindo espaço para implementações de algoritmos que possam levar a automatizações de funções de gerenciamento de rede.

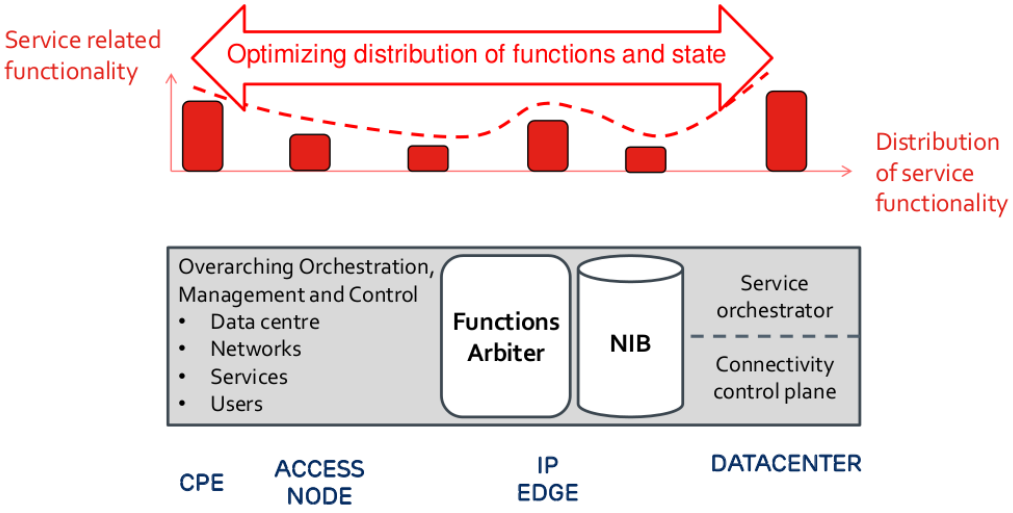


Figura 2.8: Visão lógica de gerenciamento (CSáSZáR et al., 2013).

Um framework de NFV pode requerer alocações dinâmicas de recursos de *hardware* conforme a carga de operação de NFs sobre ele, o que deve tornar a tarefa de orquestração flexível (*scale up - scale down*). A Figura 2.8 apresenta visões amplas de rede que devem ser fornecidas ao controle de gerência e orquestração da rede para que a distribuição de NFs, bem como das cadeias de serviços que elas constituam, estejam otimizadas em suas funcionalidades e de acordo com o ambiente em que se encontram, podendo este ser homogêneo ou heterogêneo (SIRACUSA; SALVADORI; RASHEED, 2013). Neste caso, sendo estas funções desempenhadas para escalar em ambientes desconexos, a migração de NFs tem como desafio a manutenção de seu estado durante sua realocação, replicação e qualquer tipo de gerenciamento de escalabilidade. Este requisito, traz consigo a necessidade de mecanismos de coordenação e arbitragem em requisições de aplicações a provedores de infraestrutura, como em (SAMDANIS et al., 2013). Além disso, para que estes estados sejam corretamente implementados, formas de checagem de configurações designam papel crucial na determinação de consistências de NFs que permitirão o cumprimento dos requisitos de desempenho e escalabilidade.

Junto aos serviços de provimento dinâmico de capacidade das NFs e checagens de consistências destas operações, estão as tarefas de detecção, diagnóstico e recuperação de falhas. Cabe a função de gerenciamento verificar se uma infraestrutura de NFV está operacional em seus diversos níveis. Nesse caso, tal monitoramento deve estar atrelado a cumprimentos de contratos de SLA fim-a-fim que podem levar em consideração requisitos de tempo, espaço e custo.

2.5.4 Portabilidade e Interoperabilidade com Plataformas Legadas

Diante da prevista evolução contínua de implementação dos conceitos de NFV em redes de produção, a interoperabilidade com redes legadas se torna um ponto chave onde a possibilidade de coexistência entre VFs e PFs estará amplamente atrelada às interfaces de comunicação disponíveis a estas funções, como por exemplo, em tunelamentos utilizando SDNs (KOBAYASHI et al., 2013). Nesse aspecto, uma visão ampla de rede possibilitada por plataformas de gerenciamento podem definir cadeias de serviços constituídas por *middleboxes* dedicadas e NFs em equipamentos de *hardware* genéricos. A verificação do impacto de cadeias de serviços de plataformas legadas em novas funções de rede e vice-versa é um desafio que pode ser avaliado com base nas condições de construção de cadeias de serviços como mencionado na subseção 2.5.1.

O desenvolvimento de sistemas de comunicação espontâneos para interoperabilidade dinâmica e universal entre sistemas legados e novas tecnologias tem as seguintes dificuldades quando determinados em uma única e padronizada solução: formular um sistema desta forma que disponha de maneira *one size fits all* pode não conseguir lidar com a heterogeneidade das diversas tecnologias existentes em ambientes NFVI; padrões de desenvolvimento e comunicação são lentos e construídos em processos incrementais enquanto que novas aplicações e sistemas distribuídos surgem constantemente e podem tornar as soluções existentes rapidamente obsoletas; plataformas legadas continuam úteis aos serviços atuais, enquanto que novos padrões nem sempre levam em consideração questões de interoperabilidade com soluções antigas, ainda operacionais.

Como requisito de operação de NFs, é necessário na visualização de informações sobre a rede tornar transparente, por parte do operador de rede ao cliente, a utilização ou não de VFs ou PFs bem como o impacto de desempenho causado no serviço requisitado pela utilização destas funções (GS, 2013c). Tal fato, implica em outro desafio a ser avaliado na determinação de cadeias de serviços fim-a-fim onde múltiplos provedores de NFVI intermediários possuem diferentes tecnologias de VNFs necessitando de interoperabilidade, não só das funções de rede em si, mas também de negociações de parâmetros de SLAs nas diversas cadeias heterogêneas de

serviços que serão constituídas entre eles. Por este aspecto, necessitam ser especificadas às tecnologias utilizadas ao longo do serviço fim-a-fim constituído, por exemplo, se houve utilização de comutação de pacotes ou estabelecimento de circuitos ópticos. Além disso, composições de redes legadas e novas tecnologias envolvem parâmetros diversificados de controles de procedimentos, protocolos, consumo de energia, entre outros parâmetros de rede (ex: uso de largura de banda e latência em operações) as quais devem ser dimensionadas corretamente para atender as demandas de cadeias de serviços fim-a-fim (SIRACUSA; SALVADORI; RASHEED, 2013).

2.5.5 Estabilidade da rede: Segurança e Resiliência

Da mesma forma que a existência de virtualização em servidores cria brechas para o surgimento de vulnerabilidades em suas camadas de operação (ex: OS, hipervisores, *hardware*), a determinação de funções de rede sobre um conjunto de servidores e *switches* COTS ou definidos pelo modelo BYOD também podem propiciar o surgimento de falhas de segurança em novos níveis de virtualização advindos de NFV, tais como sistemas operacionais de rede, cadeias de serviços de VNFGs e interfaces entre NFs (GS, 2013b). Dessa maneira, em uma infraestrutura de NFVI, assim como em data centers com milhares de servidores, diversos provedores de serviços e operadores de rede podem utilizar serviços compartilhados, e requisitam como principal funcionalidade o isolamento de desempenho entre os contratantes de fatias do serviço a fim de que não tenham seus contratos de SLA violados.

Clientes requisitando operações de gerenciamento de NFs ou de NFVIs podem requerer controle sobre os serviços que contratam, e logo, possuir a necessidade de gerenciá-los em seus diferentes níveis de operação. Em seu pleno funcionamento, NFV deve fornecer a clientes a exposição segura de APIs com as quais se possa gerenciar NFs de acordo com os níveis de serviço contratados. Consequentemente, a estas APIs não devem ser dadas as permissões de clientes constituirem NFs ou cadeias destas que interfiram no funcionamento de NFs ou serviços de outros clientes e, portanto, da própria NFVI. Exemplo este encontrado em proposta de caso de uso de plataforma de rede virtual como serviço (*Virtual Network Platform as a Service* - VNPaaS) na atribuição de políticas de segurança em *firewalls* para que empresas implementem aplicações de acesso a internet através de provedores de serviços (GS, 2013a) ou requisitos de segurança já existentes em SDN (SCOTT-HAYWARD; O'CALLAGHAN; SEZER, 2013), a serem futuramente utilizados em funções de orquestração de cadeias de serviços em NFV.

Entre outros fatores, como o grande impacto que políticas de alto nível causam atualmente a *middleboxes* e suas respectivas funções de rede, em um ambiente de NFV cadeias de serviços fim-a-fim trarão consigo a presença de inúmeras determinações de cumprimentos de SLAs.

Adicionalmente a este fato, as cadeias de políticas que estas poderão definir, serão de grande relevância, e de certa forma mais importante do que os próprios requisitos de segurança de NFs e suas interfaces. A determinação de consistências em políticas de alto nível podem levar funções de gerenciamento a operar de acordo com requisitos de provedores de serviço de modo que elas desempenhem papel fundamental no uso correto de NFVIs. Portanto, tanto em redes legadas quanto em novas tecnologias que irão surgir de PoCs de NFV, a constituição de políticas de segurança de alto nível é um fator preponderante na existência de funções de orquestração e gerenciamento consistentes a atender contratos de SLA de forma segura para clientes e para os próprios provedores de infraestrutura (GS, 2013c).

Quanto a resiliência, é considerável que em um ambiente de NFV exista requisitos de determinação de recriação de NFs caso estas venham a falhar. Tal recriação pode ocorrer de modo automático ou manual, conforme requisitos de operação do serviço dependente dessas NFs. Além disso, para que esta tarefa seja possível de ser realizada, a determinação de níveis de requisitos de confiabilidade/disponibilidade devem ser feitos para o agrupamento de VNFs em diferentes categorias de resiliência. Consequentemente, a determinação de funções necessárias à continuidade dos serviços constituídos por orquestração de NFVs devem facilitar aos planos de controle e de dados, formas seguras de disponibilidade e continuidade de serviços e não se tornarem um ponto único de falhas (GS, 2013c).

Não obstante, NFVs devem permitir que seus dados possam ser replicados para preservação de suas integridades com a devida performance necessária ao cumprimento de SLAs. Estas devem definir métricas que determinem valores e a variabilidade de padrões de estabilidade de VNFs, inerentes a determinação das características de confiabilidade de um framework NFV (GS, 2013b). Nesse aspecto, diferentes métricas podem ser estabelecidas, tais como: taxa máxima de perdas de pacotes não intencional; variação máxima de atraso e latência baseada em fluxo de dados; tempo máximo para detectar e recuperar falhas; e taxa máxima de falhas de transações válidas e que não invalidam outras transações. Estes parâmetros podem variar bastante conforme a determinação de serviços específicos (ex: voz e vídeo, transações financeiras, aplicações relacionadas a saúde) que podem requerer confiabilidade edisponibilidade maior do que SLAs de melhor esforço.

Um tópico interessante no que diz respeito a constituição de cadeias de serviços é a determinação de suas falhas e correlações entre elas e as diferentes NFs que as constituem, sejam elas físicas ou virtuais, como no caso de uso de uma infraestrutura de NFV como serviço (GS, 2013a). Para isso, todo o potencial da flexibilidade provida pela virtualização e portabilidade de funções de rede deve ser utilizada para atingir a confiabilidade necessária a disponibilidade e continuidade

de serviços. Esta tarefa pode ser realizada pela definição clara de interfaces entre cadeias de serviços, estimativas bem definidas de desempenho de NFs, técnicas de verificações constantes de SLAs, checagem de dependência entre NFs, e por fim, formas de atribuição de correlações entre falhas conforme a determinação de conjuntos de NFs em grupos semelhantes de requisitos de SLA.

2.5.6 NFV e SDN

Redes Definidas por *software* (SDNs) (FEAMSTER; REXFORD; ZEGURA, 2013) se baseiam na separação dos planos de dados e de controle da rede, sendo que este se refere ao conjunto de funções, logicamente centralizado em controladores de rede, que influencia em como os pacotes são encaminhados a destinos na rede por elementos que aquele define para realizar tal tarefa por meio de uma interface de comunicação bem definida. Dessa forma, a inteligência da rede se concentra em sua maior parte no plano de controle, o qual pode potencialmente abrigar qualquer aplicação de rede que possibilite a implementação de melhores estratégias para encaminhamento de tráfego por inúmeros atuadores em diferentes granularidades. Consequentemente, SDN pode estabelecer algoritmos eficientes no plano de dados para atuar no balanceamento de carga em enlaces, tendo como critérios políticas que contenham quaisquer critérios que sejam úteis a esta tarefa.

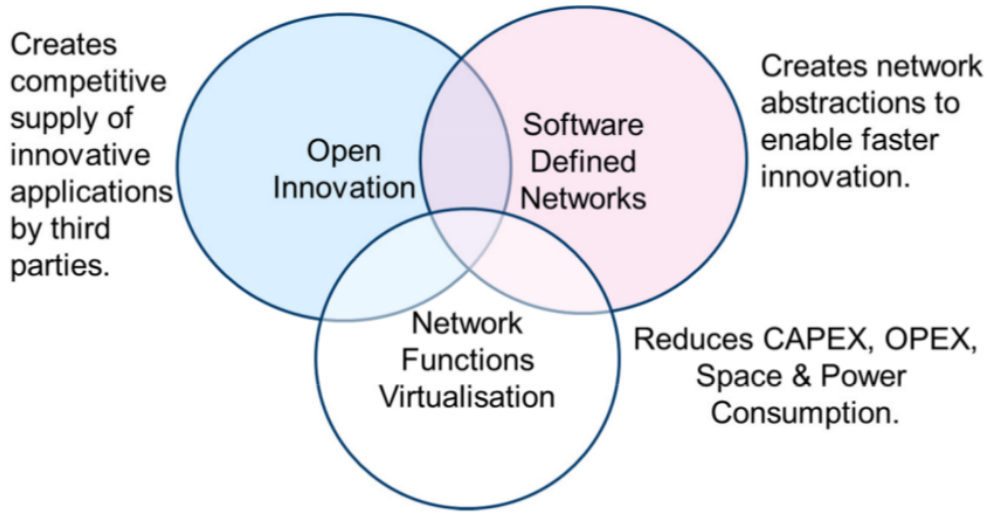


Figura 2.9: Relação entre SDN e NFV (NFV White Paper, 2012).

É muito importante ressaltar que SDN e NFV são independentes, mas podem ser complementares principalmente nas tarefas de gerenciamento (KIM; FEAMSTER, 2013) e orquestração (ONF, 2014). Figura 2.9. A customização de funções de rede, bem como de suas programações no plano de dados, carecem de atenção no que diz respeito a permissividade de agentes, se-

jam estes usuários finais ou operadores de rede. Isto pois, segundo requisitos de portabilidade de NFV, tais agentes podem requerer a instalação de aplicações e funções de rede em dispositivos genéricos (ex: BYOD) de modo semelhante ao que existe na computação, em sistemas operacionais. Dessa maneira, a padronização de interfaces, *northbound* e *southbound*, para programabilidade de rede por parte de SDN, necessitam ser estipuladas, seja por *hardware* ou *software* em diferentes linguagens de baixo e alto nível (ex: Python, Java, Procera, Pyretic), para que a orquestração e gerenciamento da rede defina NFs, e consequentemente, VNFGs, como proposto em (PENTIKOUSIS; WANG; HU, 2013) e (JIN et al., 2013) na programação de caminhos em núcleos de rede sem fio, por exemplo. Estas interfaces, ao mesmo tempo que facilitam a diversificação de aplicações e utilização por parte de agentes também podem prejudicar modelos de desenvolvimento e de negócios, pois, a evolução de diferentes plataformas de *hardware* e *software* por diferentes fabricantes de *hardware* podem ser limitadas caso a padronização de interfaces não possibilite formas de diferenciação de produtos, sejam eles controladores de rede ou dispositivos BYOD (RISSO; MANZALINI; NEMIROVSKY, 2013).

Diante dos atuais investimentos em padronização de interfaces *northbound* do plano de controle pela *Open Networking Foundation* (ONF), algumas críticas se apresentam devido ao fato desse esforço ter um escopo tão abrangente, que tanto a programabilidade atualmente almejada em SDNs quanto o atual modelo de negócio em redes de comunicação não se adequam as atuais tecnologias habilitadoras a um plano de dados coerente às atuais demandas de escalabilidade e desempenho (ex: balanceamento de carga, IPS). Consequentemente, tais críticas se empregam à NFV, pois características de orquestração e gerenciamento, principalmente no que diz respeito a criação de cadeias de serviços, não encontram características suficientes que as habilitem a ser implementadas na atualidade (SEZER et al., 2013). Além disso, ainda existem incertezas sobre quais funcionalidades realmente devem ser separadas nos planos de dados e controle, que ainda decorrem principalmente em problemas de escalabilidade e desempenho.

Capítulo 3

ARQUITETURA PROPOSTA

Este capítulo discorre sobre a arquitetura proposta neste trabalho. A Seção 3.1 detalha os requisitos inerentes a NFV que serviram de base para definição da arquitetura. A Seção 3.2 detalha a arquitetura proposta, enquanto a Seção 3.3 apresenta a metodologia utilizada como prova de conceito.

3.1 Requisitos para Escalabilidade Eficiente de Recursos NAT em NFV

De forma a possibilitar o oferecimento da VNF NAT nas condições desejadas, atingindo a escalabilidade eficiente, alguns requisitos devem ser previamente elencados e discutidos. Desse modo, esses irão nortear o processo de decisão da arquitetura e experimentos.

3.1.1 Requisitos gerais do NFV

Inicialmente, como descrito na seção 2.5, observamos que dentre os requisitos gerais do NFV, são relevantes a garantia de baixo custo da implantação e facilidade de obtenção e manutenção de todo ambiente. Tais requisitos são atendidos através da utilização de componentes de hardware de uso geral, garantindo maior facilidade de aquisição devido ao custo estável -às vezes com baixo custo-, existência de grande quantidade de centros de distribuição e maior disponibilidade de mão-de-obra qualificada; além do emprego de *software* sem restrições por questões de autorização de uso ou necessidade de pagamento de licenças. A fim de atender esses requisitos, definiu-se a utilização de servidores de uso geral com um sistema operacional de uso livre, o Linux, como base da arquitetura e prova de conceito.

3.1.2 Requisitos específicos da VNF NAT

Quanto ao serviço de NAT, este é largamente utilizado em ambientes concentradores de acesso a Internet, como por exemplo os provedores de serviços de Internet (PSI), onde normalmente há uma grande quantidade de requisições do serviço, acarretando um alto número de traduções de endereços que devem ser devidamente gerenciados pela infraestrutura de rede.

Neste tipo de ambiente *carrier-grade NAT*, todo gerenciamento das traduções é realizado em tempo de execução. Além disso, ele deve suportar até milhares de traduções simultâneas, baseado em uma mistura de fluxos, desde transferências de grandes volumes de dados até solicitações recorrentes e breves. Desse modo, é preciso garantir e controlar o uso eficiente de memória, em função das traduções usarem caches rápidos em memória. Bem como, controlar o poder de processamento da máquina hospedeira executando a VNF. Esse controle deve ser feito para minimizar o consumo de recursos por instância virtual, viabilizando a escalabilidade e resiliência do ambiente (GS, 2013c).

Uma grande variedade de sistemas virtualizados possuem a função de rede NAT, como por exemplo o Linux virtualizado, através do programa *Iptables*, ou o roteador virtual *Vyatta*[1]. Entretanto, existem diversas limitações nessas abordagens, por exemplo, o uso de uma máquina virtual completa para realizar NAT tem alta sobrecarga. Por outro lado, os *containers*, uma tecnologia de isolamento da pilha de rede por processo, é uma abordagem de virtualização leve, porém de dificil controle de vazão e também com reuso do mesmo kernel. Dessa forma, dentre as tecnologias, definiu-se o sistema minimalista baseado em ClickOS, detalhado na Seção 4.1, como instância base para a VNF NAT da arquitetura proposta, pois, ele é capaz de atingir desempenho satisfatório, isolamento, mínimo consumo de memória da máquina hospedeira, além de ser baseado em *software* de uso livre (MARTINS et al., 2013).

3.1.3 Requisitos de controle da escalabilidade da VNF

A implantação de sistemas de controle envolvendo sistemas computacionais deve obedecer requisitos específicos de duas abordagens diferentes.

A primeira corresponde aos Sistemas de Controle em Rede (NCS), onde a troca de informações (variável de controle, variável de processo, erro, valor de referência, etc.) é realizada através de componentes de sistemas de controle (sensores, controladores, atuadores, etc.) utilizando uma rede compartilhada. Este tipo de implementação tem como requisitos o detalhamento e relação de interferência ocasionado pelas tecnologias de rede envolvidas, atraso em redes, alocação de

[1] http://www.brocade.com/en/products-services/*software*-networking/network-functions-virtualization.html

recursos, agendamento, segurança em redes de tempo real, integração de componentes físicos de controle com a rede, tolerância a falhas, entre outros (GUPTA; CHOW, 2010).

A outra abordagem é quanto a utilização do sistema para controle centralizado dos próprios recursos computacionais. Nessa abordagem, são requisitos a definição de quais recursos serão controlados (espaço em disco ou memória, poder de processamento, vazão de rede, etc.) e como eles serão controlados (limitação do recurso físico, migração de instâncias virtualizadas, controle lógico, entre outros). Além disso, também deve ser definido o algoritmo de controle utilizado entre as muitas opções existentes, como as muitas variações do controlador Proporcional Integral Derivativo (PID) (ÅSTRÖM; HÄGGLUND, 1995), Lógica Fuzzy (MENDEL, 1995), entre outros.

Com base nos requisitos do controle de escalabilidade da VNF, definiu-se o uso da segunda abordagem, de modo a optimizar a utilização da estrutura computacional que garanta a segregação do processamento e tráfego de rede comum ao do controlador, evitando interferências. Nesse trabalho, o controlador modelado é baseado no algoritmo PI (Proporcional Integral), que é bastante utilizado na literatura de redes (ex: (COUTO; CAMPISTA; COSTA, 2014)) e com baixa complexidade de implementação.

3.2 Arquitetura

De forma a atender aos requisitos do ambiente para suporte à escalabilidade eficiente da VNF NAT, definiu-se um modelo de arquitetura conforme apresentada na Figura 3.1.

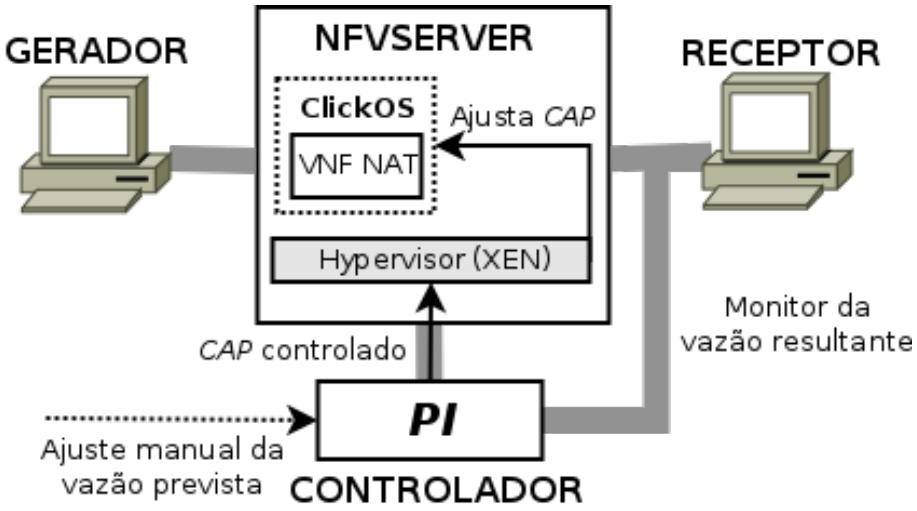


Figura 3.1: Arquitetura para suporte à escalabilidade eficiente de recursos NFV.

Nessa arquitetura modelamos os nós de redes “anteriores” e “posteriores” à tradução NAT

do tráfego, como GERADOR e RECEPTOR, de modo a simplificar o entendimento, e também conduzir os experimentos de validação. O modelo permite identificar melhor as ações do servidor NFV (NFVSERVER), pois todo o tráfego multiplexado de origem fica simplificado em GERADOR e esse tráfego deve passar pelas instâncias de VNFs NAT baseadas em ClickOS, que estão hospedadas no NFVSERVER. O elemento CONTROLADOR pode estar co-posicionado na mesma máquina ou separada. Ele monitora a vazão de saída agregada do NFVSERVER ao RECEPTOR, verificando se a vazão está de acordo com a prevista. Essa vazão é configurada previamente por uma fonte externa, por exemplo, através de automatização de parametrizações baseados em critérios como sazonalidade, número de traduções nos últimos segundos, utilização total de CPU e custo. Esse ajuste, de acordo com uma diversidade de critérios, permite o controlador atuar junto ao NFVSERVER ajustando o poder de processamento das instâncias da VNF a fim de aumentar ou diminuir a vazão da VNF.

O principal elemento da arquitetura é o NFVSERVER, onde estão hospedadas diferentes instâncias do serviço NAT virtualizado no sistema minimalista ClickOS. Nessa estrutura, o CONTROLADOR é responsável por definir um valor de controle, através de equações do modelo PI, comunicando-se por meio de API baseada nos comandos do hipervisor Xen com o NFVSERVER, de modo a controlar a vazão da VNF através de ajustes do poder de processamento da instância ClickOS com o controle do *cap* da instância. Tal ajuste permite aumentar ou diminuir a vazão suportada pela VNF de acordo com o poder de processamento atribuído. Neste cenário, o hipervisor Xen então atua sobre o processamento de escalonamento da CPU e a vazão muda. Esse ciclo com retroalimentação ocorre constantemente, evitando que oscilações causadas pelo GERADOR sejam propagadas ao RECEPTOR, portanto, estabelecendo um processo de ajuste suave ao longo da vida do NAT virtualizado.

3.3 Implementação e Análise de Desempenho

Como prova de conceito da arquitetura NFV de baixo custo, nos capítulos 4 e 5 apresentaremos uma discussão dos componentes chave implementados de nossa proposta. No Capítulo 4 é detalhado a utilização do ClickOS como sistema minimalista de alto desempenho como base para a VNF NAT. No Capítulo 5 é implementado o mecanismo de controle e auto-escalabilidade da vazão através de um sistema de controle do tipo PI. Nessa discussão serão apresentados experimentos e resultados que demonstram a escalabilidade, controle suave e o desempenho da VNF NAT baseada em instancias do ClickOS.

Capítulo 4

SISTEMA MINIMALISTA COMO BASE DA VNF

Este capítulo detalha a utilização do ClickOS como base da VNF NAT. Para isso, na Seção 4.1 são detalhados a estrutura e funcionamento do ClickOS, enquanto na Seção 4.2 são apresentados o ambiente experimental, valores prévios definidos para alguns parâmetros e variáveis do ambiente, além do detalhamento dos experimentos e apresentação dos resultados da saturação de vazão do ClickOS como base da VNF e vazão atingida versus cap ajustado do processador.

4.1 ClickOS

Para que a virtualização de funções apresente os resultados esperados, é necessário que as VMs responsáveis pelas NFs sejam concebidas seguindo algumas características importantes (MARTINS et al., 2013). São elas:

- **Rápida Instanciação:** o sistema deve ser capaz de instanciar rapidamente as NFs onde for necessário, acompanhando mudanças causadas pelo SDN;
- **Economia de Recursos:** o sistema deve suportar grande quantidade de NFs no mesmo servidor, evitando necessidade de novas compras ou custos operacionais;
- **Isolamento:** a segregação de fatias de rede deve ser garantida para que NFs de um usuário não comprometam outras NFs, tanto do ponto de vista de segurança como desempenho;
- **Desempenho:** ambientes de NFs devem possuir desempenho no plano de dados compatível com ambientes de hardware dedicado;
- **Flexibilidade:** o sistema deve suportar um grande número de NFs e ser escalável.

Considerando estas características, é possível identificar que a virtualização de recursos instanciados em VMs Linux, por exemplo, não é a melhor solução ao NFV. Esse tipo de virtualização não atende grande parte das premissas apontadas anteriormente, pois, apesar de proporcionar isolamento e bom desempenho, também apresenta alto consumo de memória e espaço em disco, além de um tempo de instanciação elevado.

Em contrapartida, existem tecnologias flexíveis e que apresentam alto desempenho, como o caso do *Click Modular Router* (Click)[1], porém, ele não garante o isolamento perfeito e ainda apresenta um alto consumo de recursos de hardware por ser processado a nível de kernel do Linux (KOHLER et al., 2000).

Buscando sanar essas deficiências, Martins et al. (2013) desenvolveu o ClickOS, uma VM leve baseada no Xen que suporta o Click. Devido às otimizações realizadas nessa VM e na estrutura de rede do próprio Xen, o ClickOS alcançou as seguintes características principais:

- **Rápida Instanciação:** cerca de 30 milisegundos;
- **Economia de Recursos:** tamanho da imagem de 1,4MB até 5MB, quando instanciado;
- **Isolamento:** garantido pelo Xen;
- **Desempenho:** até 10Gbps de vazão;
- **Flexibilidade:** suporta grafos diversos do *Click Modular Router*.

Essas características mitigam limitações impostas pela virtualização convencional ao NFV.

Segundo Martins et al. (2013), durante a construção do ClickOS, foi observado que muitos serviços oferecidos pelo kernel do Linux não eram necessários ao Click, afinal, ele suportava apenas uma única função em execução por instância. Tal característica possibilitou a simplificação do ClickOS com a retirada do suporte a múltiplos usuários, separação entre espaço de usuário e de kernel, e chamadas de sistema. Além disso, o Click não precisava de múltiplos espaços de endereçamento de memória, pois, uma única configuração é executada no mesmo espaço de endereçamento, com seus *Elementos* apenas passando ponteiros entre eles para a execução de suas funcionalidades, implicando a não necessidade de suporte a vários processos. *Threads* eram necessárias, pois, diferentes cadeias de processamento podem ser executadas em

[1] *Click Modular Router* é uma arquitetura modular de *software* responsável pela construção de *middleboxes*. Cada bloco formador do Click é denominado *Elemento* e representa uma unidade de processamento. A conexão de um conjunto desses *Elementos* em forma de grafo corresponde à configuração do *middlebox*, representando o conjunto de funções desempenhadas por ele. A execução dessas funções decorre do movimento dos pacotes de um elemento para o outro através das arestas do grafo.

paralelo. Para o Click, o acesso à rede pode ser suportada com *drivers* genéricos, deixando a complexidade do tratamento de *drivers* de hardwares específicos ao hipervisor. Sistemas de arquivo são raramente utilizados, bem como outras formas de entrada e saída, como vídeo, usb etc.

Tudo isso facultou ao Click ser executado em um ambiente muito mais simples do que VMs convencionais, o que possibilitou a formação de sua estrutura através da implementação de um grafo do Click em um OS minimalista suportado pelo Xen (MiniOS). Figura 4.1.

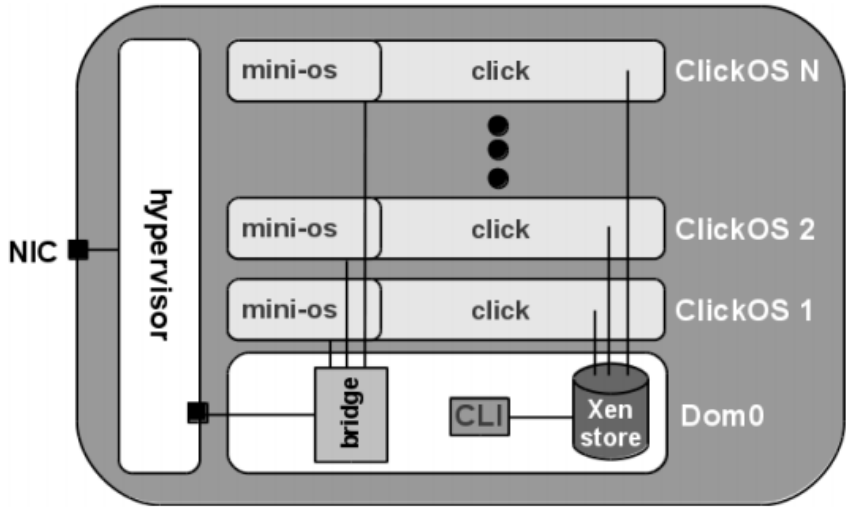
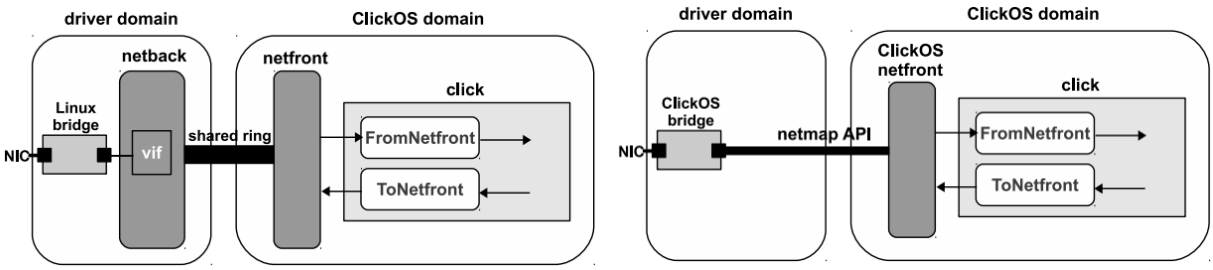


Figura 4.1: Arquitetura ClickOS (MARTINS et al., 2013).

O MiniOS suporta paravirtualização, garantindo todos recursos necessários ao Click sem os excessos apresentados anteriormente. Mais detalhadamente, o MiniOS implementa o básico para operar no ambiente Xen, possuindo tabelas de acesso a memória compartilhada entre os domínios (Dom0 e DomU); *drivers* paravirtualizados de rede (*netfront*) e armazenamento; canais para comunicação de eventos; espaço único de endereçamento; não há separação entre usuário e kernel; presença de um agendador cooperativo, reduzindo os custos de mudança de contexto.

Na implementação do ClickOS, diversas adequações que visavam melhorar seu desempenho foram realizadas tanto no Click quanto no MiniOS. Na rede, por exemplo (Figura 4.2), o Xen originalmente possui um modelo de *driver* de rede dividido: a parte externa da rede (*netback driver*), executada no domínio de *driver*, comunica-se com a NIC e exporta uma API baseada em anel; e a parte interna (*netfront driver*), executada no domínio da VM ClickOS, comunica-se com o *netback driver* via memória compartilhada (o anel). Este modelo permite que a VM ClickOS acesse a NIC do servidor mesmo sem possuir seu *driver*. A comunicação entre a NIC e a interface virtual de rede da VM (*Virtual Interface* - vif) é feita através de *bridges* Linux ou *Open vSwitch*. Figura 4.2a. Nessa implementação, os pacotes recebidos são enca-



(a) Rede Básica ClickOS no Xen.

(b) Rede Otimizada ClickOS no Xen: driver netfront modificado e rede back-end usando API Netmap e switch VALE melhorado.

Figura 4.2: Rede Básica e Otimizada ClickOS no Xen (MARTINS et al., 2013).

minhados para a *vif* de endereço MAC correspondente ao destino, que os enfilera no *netback driver*. Nesse momento, alguma *thread* do *netback driver* os coloca na memória compartilhada, notificando o *netfront driver*.

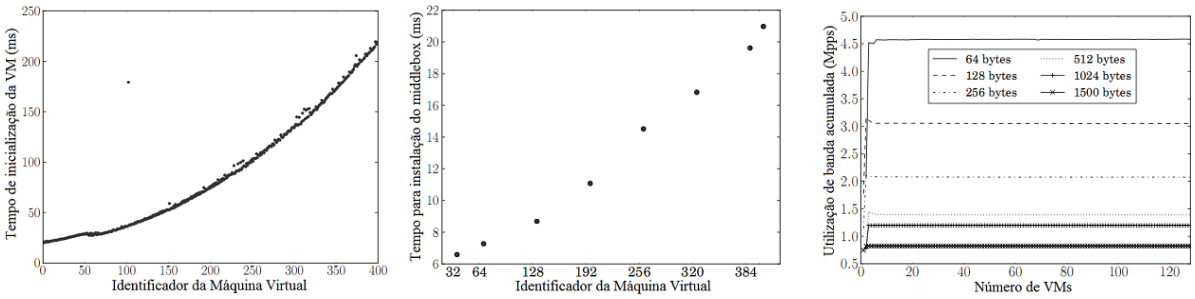
Para possibilitar o Click interagir com o *netfront driver*, Martins et al. (2013) criaram dois novos *Elementos*: *FromNetfront* e *ToNetfront*. O primeiro é responsável por inicializar a interface de rede e recuperar os dados do *netfront driver*, já o *ToNetfront* aciona a função de envio de dados do *netfront driver*.

Nesta etapa do desenvolvimento do ClickOS, testes realizados por Martins et al. (2013) apresentaram desempenho baixo, atingindo aproximadamente apenas 1% de utilização de banda num servidor com interface específicas de 10Gbps.

A Figura 4.2b apresenta as adequações feitas posteriormente no ambiente. Nesse caso, as *vifs* foram abolidas e o *netback driver* foi substituído pelo *Netmap*, proporcionando um caminho mais direto entre a NIC e a VM. O *Open vSwitch* foi substituído pelo switch virtual VALE com suporte ao *Netmap*[2] (RIZZO, 2012). O *netfront driver* foi otimizado com a introdução de dois novos mecanismos: troca do mecanismo de recebimento baseado em interrupção por um baseado em consultas periódicas; e reutilização da permissão de compartilhamento de memória das interfaces entre domínios durante toda existência do dispositivo. Quanto ao armazenamento das imagens ClickOS, estas passaram de dispositivos de armazenamento em rede (*Network File System* - NFS) para armazenamento em memórias de acesso randômico (*Random Access Memory* - RAM), o que também contribuiu para a melhora do desempenho total do ClickOS.

Após todas adequações no ambiente, o tempo de criação e inicialização de uma VM ClickOS passou para 20.789 milisegundos, sendo complementado pelo tempo de instalação do Click sobre o ClickOS, 8 milisegundos. Todo esse processo totalizou um tempo de aproximadamente

[2] *Netmap* é uma estrutura de rede que suporta comutação de pacotes a taxas muito elevadas no espaço de usuário.



(a) Tempo para criação e inicialização de 400 ClickOS.

(b) Tempo de instalação do middlebox em ClickOS já inicializados.

(c) Utilização de banda acumulada em 128 ClickOS.

Figura 4.3: ClickOS: Tempos de criação, inicialização, instalação de middlebox e utilização de banda acumulada (MARTINS et al., 2013).

30 milisegundos desde o momento da criação da VM até ela estar pronta para trabalhar.

4.2 Avaliação do Sistema Minimalista como Base da VNF

Esta seção detalha a utilização do ClickOS como base da VNF NAT. Para isso, na Seção 4.2.1 são apresentados o hardware e *software* utilizados na plataforma experimental. Na Seção 4.2.2 são discutidos a definição de valores prévios para alguns parâmetros e variáveis do ambiente. A Seção 4.2.3 detalha os experimentos e apresenta os resultados da saturação de vazão do ClickOS como base da VNF. Na Seção 4.2.4 é detalhado os experimentos e apresentado os resultados referentes à vazão do ClickOS versus o *cap* do processador ajustado.

4.2.1 Ambiente Experimental

O ambiente experimental, base do desenvolvimento deste trabalho, é representado pela arquitetura apresentada na Figura 3.1. Nela, o sistema operacional utilizado em todas máquinas foi o Linux Debian versão 8, Kernel versão 4.2. A máquina NFVSERVER possui o hipervisor Xen 4.5.1 instalado, e as máquinas GERADOR e RECEPTOR possuem o programa *pktgen* responsável por gerar e receber o tráfego nos experimetos.

A Tabela 4.1 apresenta a configuração de hardware das máquinas da Figura 3.1. Nela é possível observar que foram utilizadas placas de rede Intel PCI Express PRO/1000 de 1Gbps no NFVSERVER. Alguns estudos, como (MARTINS et al., 2013), detalhado na Seção 4.1, e (OLTEANU; HUICI; RAICIU, 2015), tiveram como objetivo provas de conceitos e desenvolvimento de tecnologias que visavam atingir a maior vazão possível no ambiente e, para isso, utilizavam hardware de custo não tão reduzido, como por exemplo as interfaces de rede Intel

IXGBE de 10Gbps.

Tabela 4.1: Configuração de hardware das máquinas utilizadas nos experimentos.

Nome	Modelo	Processador	Memória	Rede
NFVSERVER	Dell PowerEdge 6850	Intel Xeon 7100, 3.16GHz, 4 núcleos (8 com *hyperthreading*)	8GB ECC DDR2	Intel PCI Express X8 PRO/1000 PT Dual Port
GERADOR	Dell PowerEdge R210 II	Intel Xeon E3-1270 v3, 3.50GHz	16GB ECC DDR3	Intel PCI Express X8 PRO/1000 PT Dual Port
RECEPTOR	Notebook Avell Titanium B155	Intel Core i7-4810MQ, de 2.80GHz	8GB HyperX DDR3	1 Realtek PCI Express Gigabit Ethernet

4.2.2 Definições e Configurações Preliminares

Antes de realizar qualquer trabalho relacionado à análise de viabilidade do ClickOS no ambiente proposto, procurou-se identificar características técnicas e comportamentais que pudessem interferir nos resultados finais.

Relacionado ao ambiente técnico, experimentos foram realizados para responder quanto ao custo de se obter o melhor desempenho do hardware utilizado versus a complexidade que determinada configuração implicaria ao ambiente, bem como se tal solução o tornaria menos compatível com outras aplicações. Nessa linha, foram analisadas as interferências causadas pela alocação de vCPU exclusivas e memória RAM ao DOM0 no NFVSERVER.

Quanto ao processamento, observou-se que o consumo de vCPU do DOM0 ficou sempre abaixo do consumo máximo de 2 vCPU, mesmo sob carga máxima entre suas interfaces de rede. Quanto a alocação de memória RAM, verificou-se que sua utilização se manteve inalterada em todos cenários. Mesmo com esses resultados, optou-se em alocar 2048MB de memória RAM e 2 vCPU com *hyperthreading* desabilitado e 4 vCPU com *hyperthreading* habilitado ao DOM0 no NFVSERVER durante os experimentos, de forma a reduzir a possibilidade de interferência nos resultados do controlador durante os experimentos.

Quanto à VNF NAT, esta teve como base instâncias ClickOS e Linux devidamente hospedadas no NFVSERVER. A Tabela 4.2 apresenta as configurações de hardware virtualizado alocado a cada tipo de instância. Os valores utilizados foram definidos a partir dos requisitos mínimos sugeridos para cada tipo de sistema, onde, uma única instância do Linux contou com 112MB de memória RAM e 680MB de espaço em disco reservados. Já o ClickOS contou com apenas 12MB de memória RAM e 8MB de espaço em disco reservados.

As informações contidas na Tabela 4.2 permitem observar que cada instância Linux aloca

Tabela 4.2: Hardware virtual das instâncias de VNF NAT.

Instância	vCPU	Memória	Disco	Rede
Linux	1	112 MB	680 MB	2 interfaces conectadas às bridges do DOM0
ClickOS	1	12 MB	8 MB	2 interfaces conectadas às bridges do DOM0

quase 10 vezes mais recursos de hardware da máquina hospedeira se comparado às instâncias ClickOS. Essa informação por si só já representa uma grande vantagem na utilização de VNF baseada em instâncias ClickOS.

Para análise comportamental do ambiente, inicialmente foram realizados experimentos visando medir a saturação de vazão, e vazão máxima obtida versus *cap* ajustado, utilizando-se instâncias Linux com suporte à NAT pelo módulo de mascaramento do programa *Iptables*, posteriormente, comparando aos resultados dos mesmos experimentos realizados em instâncias ClickOS com suporte à NAT.

Todos os experimentos basearam-se no envio de 3 sequências de 10 milhões de segmentos UDP de tamanho 64 Bytes cada, com origem o programa *pktgen* na máquina GERADOR e destino também o programa *pktgen* na máquina RECEPTOR. Os resultados apurados correspondem à média das 3 sequências de envio de dados.

4.2.3 Saturação da vazão

Este experimento mensura a vazão máxima da VNF baseada em instâncias Linux e ClickOS, sob tráfego de múltiplos de 20Kp/s, iniciando em 20Kp/s e finalizando em 200Kp/s. Foram realizadas simulações com emissão de cada faixa de taxa com a VNF suportando 1, 500 e 1000 endereços para traduções, sempre utilizando 50.000 endereços para emissão do tráfego.

Como exemplo, considere que a máquina GERADOR está gerando tráfego a uma taxa constante de 20Kp/s com 50.000 endereços de origem. Ela seleciona o primeiro endereço do grupo de endereços de origem, envia 10 segmentos, seleciona o segundo endereço de origem, envia mais 10 segmentos, repetindo esse procedimento 10 milhões de vezes. A lista de endereços de origem dos segmentos é reiniciada quando atingido seu fim. Para cada segmento com um novo endereço de origem que chega à VNF, há necessidade de se acrescentar uma nova tradução na tabela NAT da VNF. Por sua vez, em cada cenário, a VNF mascara a origem do tráfego com apenas 1 endereço, um conjunto de 500 ou 1000 endereços. No caso dos conjuntos de 500 ou 1000 endereços de mascaramento, a cada novo endereço de origem de tráfego que chega à VNF, o mascaramento é realizado com um novo endereço do conjunto de endereços de mascaramento, quando esse conjunto atinge seu final, inicia-se o mascaramento através das portas de

serviço.

A Figura 4.4 apresenta os resultados da saturação da instância Linux versus a instância ClickOS. Os resultados foram obtidos com o *hyperthreading* habilitado, porém, a execução dos mesmos experimentos com o *hyperthreading* desabilitado na máquina NFVSERVER apresentaram diferença desprezível na vazão da VNF.

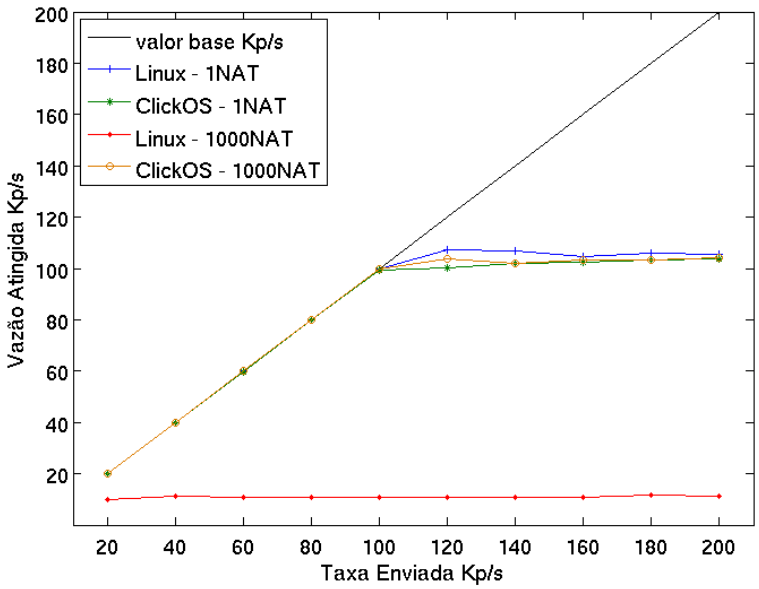


Figura 4.4: Saturação de vazão em instância Linux (a) e ClickOS (b). Taxa de transmissão em múltiplos de 20Kp/s; 50.000 endereços emissores; VNF suportando 1 e 1000 endereços de mascaramento; *hyperthreading* habilitado.

Na Figura 4.4 é possível observar que a vazão máxima suportada pela instância Linux com 1 endereço de mascaramento (1NAT) é aproximadamente 110 Kp/s. Mesmo o envio de taxas superiores, como por exemplo 200Kp/s, não resultaram vazão superior a essa marca. Por outro lado, nos cenários onde a VNF possui 1000 endereços de mascaramento, a vazão manteve-se saturada próximo aos 10Kp/s.

Observamos também que a vazão máxima atingida nas instâncias ClickOS também está próxima aos 100Kp/s, porém, nos cenários onde a VNF possui 1000 endereços de mascaramento a vazão máxima atingida permaneceu próximo à vazão limite do sistema, portanto, aproximadamente 900% superior ao suportado pelas instâncias Linux. Embora não apresentado na figura, os resultados obtidos para até 500 endereços foram similares aos de 1 NAT tanto no Linux quanto no ClickOS.

Baseado nos resultados obtidos, é possível verificar que o ClickOS apresenta vazão muito similar ao Linux em ambientes de 1 até 500 endereços de mascaramento, porém, apresentou vazão aproximadamente 900% superior quando o número de endereços aumentou para 1000.

Esse comportamento demonstra seu bom desempenho e capacidade de escalabilidade, precisando alocar aproximadamente 10 vezes menos recursos de hardware.

4.2.4 Vazão versus *cap* ajustado

Nessa etapa foi analisado a relação de vazão da VNF com o ajuste do *cap*. Para isso considerou-se apenas cenários com 50.000 endereços emissores, 1 endereço de mascaramento e *hyperthreading* habilitado. Foram gerados tráfegos de vazão em múltiplos de 20Kp/s, iniciando em 20Kp/s e finalizando em 100Kp/s. O *cap* foi ajustado inicialmente em 5% da capacidade do processador, aumentando em múltiplos de 5%.

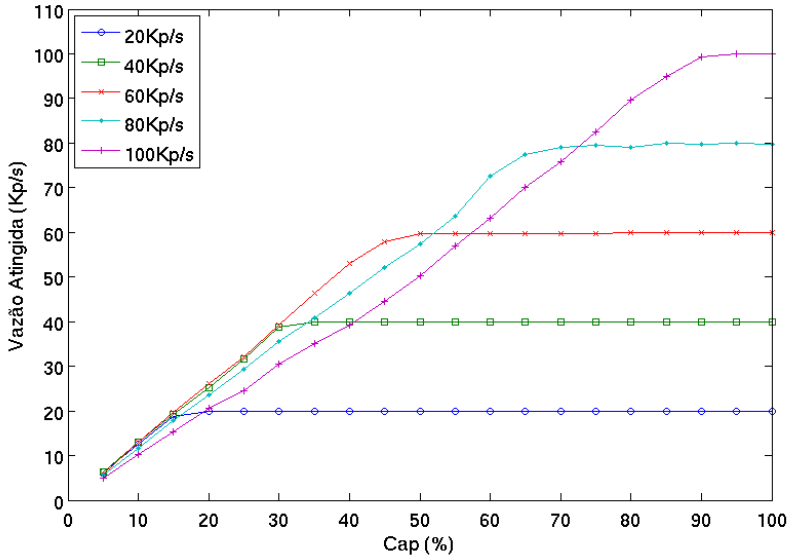


Figura 4.5: Vazão da VNF versus *cap* ajustado em instância Linux. Tráfego gerado iniciando em 20Kp/s, sendo acrescido em múltiplos de 20Kp/s. O *cap* iniciou em 5%, sendo acrescido em múltiplos de 5%.

Analisando a Figura 4.5 é possível observar que o aumento da vazão em Linux virtualizado é perfeitamente linear ao aumento do *cap*. Tal comportamento se manteve estável até a vazão da VNF atingir a taxa de transmissão enviada pelo GERADOR.

Ao comparar os resultados da Figura 4.5 com o gráfico de vazão versus *cap* de Couto, Campista e Costa (2014), é possível observar alguma diferença entre elas, porém, duas características principais devem ser consideradas: a primeira é o tipo de curva de crescimento da vazão. Ela deve-se à diferença na VNF de roteamento mais mascaramento utilizada neste trabalho, além disso, a vazão máxima da VNF foi equivalente à taxa de transmissão do GERADOR somente com a utilização de mais ciclos de processador, tal fato decorre devido a diferença na VNF e ao modelo do processador utilizado neste experimento. Mesmo assim, os resultados demonstram

que o modelo utilizado por Couto, Campista e Costa (2014) é válido para este ambiente.

A Figura 4.6 apresenta a vazão da VNF versus *cap* ajustado em instâncias ClickOS. Nela observa-se que o aumento da vazão tem um comportamento um pouco diferente, pois, o ClickOS atinge a vazão máxima do tráfego gerado apenas muito próximo ao *cap* de 100%, com utilização máxima do poder de processamento da vCPU associada à instância.

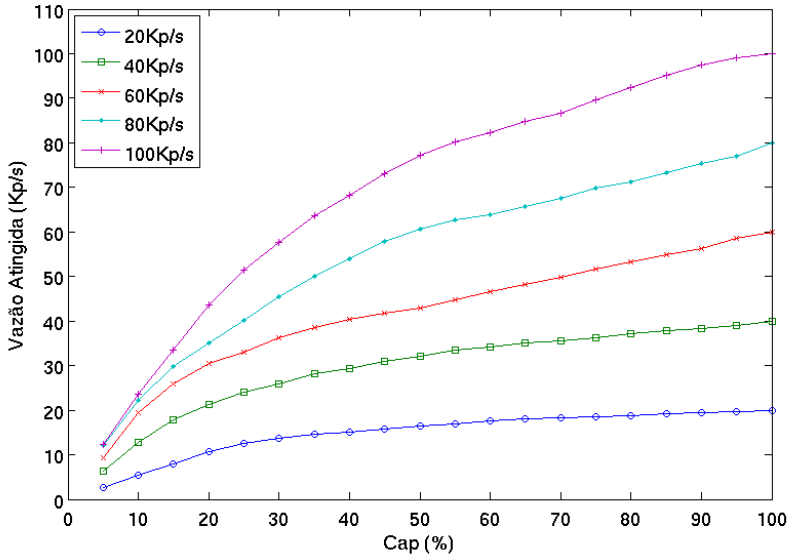


Figura 4.6: Vazão da VNF versus *cap* ajustado em instância ClickOS. Tráfego gerado iniciando em 20Kp/s, sendo acrescido em múltiplos de 20Kp/s. O *cap* iniciou em 5%, sendo acrescido em múltiplos de 5%.

Tal comportamento chamou a atenção, visto que, mesmo em taxas menores, como 20Kp/s ou 40Kp/s, por exemplo, a vazão máxima somente é atingida garantindo-se 100% do poder de processamento da vCPU ao ClickOS. Outro ponto importante é a existência de uma pequena curva no gráfico, porém, essa curva remete que o comportamento do ClickOS é compatível ao controle de vazão por ajuste do *cap*.

Considerando os resultados de saturação de vazão e vazão versus *cap* ajustado, observa-se que o ClickOS é viável para uso como instâncias base para a VNF NAT. Seu comportamento demonstra bom desempenho e capacidade de escalabilidade configurável, além de consumir pouco recurso de *hardware* da máquina hospedeira.

Capítulo 5

SISTEMA DE CONTROLE DE VAZÃO PARA VNF

Este capítulo analisa a viabilidade do emprego de teoria de controle de malha fechada para escalabilidade eficiente de VNF. Para melhor entendimento de todo conteúdo envolvido, a Seção 5.1 introduz conceitos fundamentais de teoria de controle, detalha a estrutura e funcionamento do algoritmo PID e apresenta exemplos da utilização de suas variações em sistemas de controle de redes. A Seção 5.2 detalha o emprego da variação PI do algoritmo de controle na arquitetura proposta.

5.1 Sistemas de Controle

Esta seção apresenta os principais conceitos, objetivos e metas relativos aos Sistemas de Controle (*Control Systems* - CS). Também trata do algoritmo Proporcional Integral Derivativo (PID), servindo de base conceitual para o entendimento de sua utilização no restante do trabalho.

5.1.1 Conceitos e Definições

Segundo Åström e Hägglund (1995), Goodwin, Graebe e Salgado (2000) e Ogata (2003), o desenvolvimento e a utilização de Sistemas de Controle (*Control Systems* - CS) faz parte da história humana há muito tempo. Os primeiros exemplos incluem mecanismos responsáveis por regular relógios e direcionar moinhos de vento para a direção correta dos ventos, porém, o fato fundamental para seu desenvolvimento ocorreu durante a revolução industrial, quando James Watt, desenvolveu um mecanismo de regulagem do fluxo de vapor em locomotivas.

Inúmeros outros eventos tiveram importantes contribuições para a evolução dos sistemas de controle. As guerras, com a criação de sistemas de orientação; os programas espaciais das décadas de 1960 e 1970, que contribuíram com vários avanços que posteriormente foram

aplicados a todos, são alguns exemplos de contribuições à evolução dos sistemas de controle.

Atualmente esses sistemas estão disseminados em vários elementos da sociedade moderna, apesar de, muitas vezes, não serem visíveis aos usuários finais, porém, desempenham um papel de grande importância em áreas como medicina, biologia, construção civil etc.

De maneira geral, mas não obrigatoriamente, a análise de um CS deve contemplar a seguinte estrutura:

Processo: Também chamado de *Planta*, representa o processo a ser controlado. Inicialmente defendia-se que era necessário o conhecimento físico da estrutura alvo do controle, mas com as mudanças que os processos sofreram principalmente pelo avanço tecnológico, esse conceito tornou-se mais abrangente;

Objetivos: Antes de qualquer desenvolvimento, é necessário entender quais são os reais objetivos de tal controle. Definições como: O que se quer alçancar (redução de consumo energético, melhor desempenho etc); quais variáveis precisam ser controladas para atingir esses objetivos; qual nível de desempenho é necessário (acurácia, velocidade etc), devem ser consideradas;

Sensores: São os sensores que permitem o monitoramento, possibilitando conhecer os resultados fornecidos pela estrutura controlada;

Atuadores: Enquanto os sensores monitoram os resultados fornecidos pelo ambiente permitindo a análise da atual situação do objeto controlado, os atuadores devem agir quando esses resultados estiverem fora dos valores de referência;

Comunicação: Interconecta os sensores aos atuadores. São canais de comunicação que podem ter formas e tecnologias diferentes, dependendo do ambiente controlado e CS utilizado;

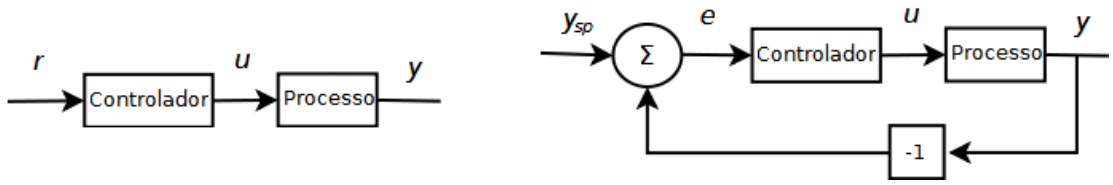
Arquitetura e interfaceamento: Questões como *o que deve ser conectado e onde* não têm resposta trivial, visto que em algumas situações pode ser interessante que todos sinais sejam levados a um ponto central para processamento enquanto em outras não. Complexidade, custo, restrições de tempo computacional, manutenção e confiabilidade são alguns outros pontos que também devem ser considerados;

Algoritmos: Parte muito importante dos CSs, correspondem à inteligência que atua entre os sensores e atuadores. São os algoritmos que tratam as informações fornecidas pelos sensores e determinam como os atuadores devem operar. Existem diversos tipos de algoritmos, cada um podendo seguir conceitos e características diferentes (ex: lógica nebulosa, rede neural, algoritmo genético etc) que determinam sua estrutura e forma de operação. A subseção 5.1.2 discute com maior profundidade o algoritmo de controle Proporcional Integral e Derivativo

(PID), utilizado no desenvolvimento deste trabalho.

Além dessas áreas apresentadas, também pode ser necessário a análise detalhada sobre outros pontos relevantes, como o envolvimento da computação nos CSs, distúrbios e incertezas envolvidos no ambiente controlado, homogeneidade, custos e benefícios da implementação do controle.

Quanto à estrutura, os CSs podem se apresentar de duas formas: Malha Aberta e Malha Fechada.



(a) Circuito de Malha Aberta - Adaptado (GOODWIN; GRAEBE; SALGADO, 2000).

(b) Circuito de Malha Fechada (ÅSTRÖM; HÄGGLUND, 1995).

Figura 5.1: Diagramas de Bloco de Sistemas de Controle de Malha Aberta e Malha Fechada.

Malha Aberta: A Figura 5.1a apresenta um diagrama de blocos de circuito de controle de Malha Aberta (*Open Loop*). Esse sistema é composto pelos componentes **Processo** e **Controlador**, representados como caixas com setas que indicam a relação causal entre suas entradas e saídas. Este é o tipo mais simples de circuito de controle, visto que o controle apenas recebe uma entrada qualquer, representada na figura como r, e realiza sobre ela os ajustes previamente programados no controlador. Já o processo recebe em sua entrada o resultado do controle, representado na figura como u. Por sua vez, a saída do processo tem como resultado o y, que é o ponto final do circuito. Esse sistema de controle é indicado em situações em que os valores de entrada do controlador (r) e sua relação sobre os valores da saída do processo (y) são previamente conhecidos e inalteráveis, possibilitando assim a inserção de parâmetros corretos no controlador para que os resultados enviados para entrada do processo (u) sejam sempre os esperados;

Malha Fechada: A Figura 5.1b apresenta um diagrama de blocos de circuito de controle de Malha Fechada (*Closed Feedback Loop*) simples. Esse sistema também é composto por dois componentes principais, o **Processo** e o **Controlador**, representados como caixas com setas que indicam a relação causal entre suas entradas e saídas. O processo tem uma entrada, a variável manipulada, também chamada de variável de controle, designada por u. O resultado do processo é chamado de variável de processo (*Process Variable* - PV), designado por y, e é medida por um sensor. O valor pretendido da variável de processo (y) é chamado de ponto de ajuste (*setpoint* - SP) ou valor de referência, sendo designado por y_{sp}. O erro de controle

é a diferença entre o valor de referência (y_{sp}) e a variável de processo (y), ou seja, e = y_{sp} - y. O controlador possui uma entrada (o erro - e) e uma saída (a variável de controle - u). A Figura 5.1b mostra que o **Processo** e o **Controlador** estão conectados por um circuito de Malha Fechada.

5.1.2 Algoritmo Proportional Integral Derivativo (PID)

Segundo Åström e Hägglund (1995), Goodwin, Graebe e Salgado (2000) e Sellers (2001), o controle PID segue o modelo de malha fechada, descrito na subseção 5.1.1, e tem sua estrutura formada por três ações: Proporcional (P), Integral (I) e Derivada (D); detalhadas a seguir:

Ação Proporcional: Responsável pela correção proporcional de acordo com a intensidade do erro proporcional (e). A intensidade da correção aplicada ao erro varia de acordo com o grau de diferença ente o valor pretendido (y_{sp}) e o resultado da variável de processo (y).

Exceto por uma condição de carga muito específica, sempre haverá diferença entre o ponto de controle e o ponto de equilíbrio para um sistema que está operando sob um controlador proporcional. Essa diferença de valores entre o valor de referência (y_{sp}) e o valor da variável de processo monitorada (y) é denominado erro proporcional (e). A Equação 5.1 apresenta a função que calcula o valor da variável de controle (u) realizado pelo controlador, com base no produto do ganho proporcional (K_P) pelo erro proporcional (e):

$$u = K_P * e \tag{5.1}$$

Onde u é a variável de controle, K_P é o ganho proporcional e e é o erro proporcional.

Como apresentado, K_P é o ganho proporcional do controlador. É ele quem determina a taxa de resposta de saída do controlador para o sinal de erro recebido.

Ação Integral: Correção proporcional à integral do produto do erro (e) pelo tempo (t) de sua duração. Mesmo pequenos erros com seguidas ocorrências devem ser corrigidos para que o sistema alcance seu ponto de equilíbrio sem erros;

A utilização da ação integral em conjunto com a proporcional, garante erro zero no ponto de equilíbrio do processo. A ação integral aplica a somatória dos erros instantâneos no cálculo da variável de controle (u), realizado pela ação proporcional no controlador, garantindo que a variável de controle (u) forneça a energia necessária ao processo para que esse mantenha-se sem erros.

$$u = K_P * e + K_I * \sum_t e \tag{5.2}$$

Onde $K_P * e$ corresponde à ação proporcional da Equação 5.1, K_I é o ganho integral do controlador e $\sum_t e$ é a integral do erro proporcional no tempo.

A ação integral, apresentada na Equação 5.2, armazena a somatória do erro instantâneo e coloca esse valor como um ganho na ação proporcional, estabilizando o sistema. Caracteriza-se também por promover uma ação lenta, visto que seu crescimento é integral em relação ao erro pelo tempo.

Ação Derivada: Correção proporcional de acordo com a predição da taxa de variação da saída do processo (y). A componente derivada faz uma predição da variação da saída do processo (y) e atua preventivamente junto ao controlador aumentando ou diminuindo sua intervenção.

A ação derivativa atua quando o processo está em estado transitório, isto é, quando a variável de processo (y) tem seu valor modificado. Nessa situação, a variável de processo (y) terá valor diferente da variável de controle (y_{sp}), modificando o erro (e) para um valor diferente de zero. Como a saída do processo (y) pode apresentar um atraso em relação ao ganho aplicado pela variável de controle (u), por decorrência de sua própria estrutura e tempo de propagação de sinal, por exemplo, a ação derivativa tenta minimizar esse atraso através da predição de erro do processo, fazendo a derivada à frente da reta tangente à curva do erro no instante atual.

Quando o erro (e) for constante (lembrando que e somente será constante em zero, devido a ação integral), a ação derivativa também apresentará resultado zero, não provocando mais efeito sobre o controlador. Sua ação ocorre apenas quando o processo está em estado transitório.

A Equação 5.3 apresenta a ação derivada, juntamente com as ações proporcional e integral:

$$u = K_P * e + K_I * \sum_t e + K_D * \partial e / \partial t \tag{5.3}$$

Onde $K_P * e + K_I * \sum_t e$ corresponde à ação proporcional e integral, apresentadas na Equação 5.2, K_D é o ganho derivado do controlador e $\partial e / \partial t$ é taxa de variação do erro em relação ao tempo t).

A Tabela 5.1 apresenta a relação entre as ações Proporcional, Integral e Derivada, apontando seus pontos de atuação no controlador e a correspondente capacidade de zerar o erro da variável de processo em seu ponto de equilíbrio:

Tabela 5.1: Ações PID e seus Pontos de Atuação e Capacidade de Equilíbio com Zero Erro.

Ação	Ponto de Atuação	Ponto de Equilíbrio
Proporcional	Intensidade do erro	Não zera o erro
Integral	Produto do erro pelo tempo de duração	Zera o erro
Derivada	Taxa de variação da saída do processo em relação ao tempo	Não zera o erro

5.1.3 PID em Sistemas de Controle Computacionais

A utilização do PID em sistemas de controle envolvendo ambientes computacionais pode ser analisado pelo menos sob duas abordagens diferentes.

Gupta e Chow (2010), por exemplo, apresentam os Sistemas de Controle em Rede (*Networked Control System* - NCS). No NCS, a troca de informações (variável de controle, variável de processo, erro, valor de referência etc) é realizada através de componentes de sistemas de controle (sensores, controladores, atuadores etc) utilizando uma rede compartilhada. Segundo os autores, NCS tem sido uma importante área de pesquisa tanto para a academia quanto para a indústria há décadas, transformando-se numa área multidiciplinar.

Em seu trabalho, Gupta e Chow (2010) apresentam as diferentes áreas de pesquisa envolvendo NCSs (como as tecnologias de rede envolvidas, atraso em redes, alocação de recursos, agendamento, segurança em redes de tempo real, integração de componentes físicos de controle com a rede, tolerância a falhas, entre outros) detalhando sua relação e interferência sobre os sistemas de controle.

Outra abordagem sobre a implementação do PID em sistemas computacionais é apresentada por Couto, Campista e Costa (2014). Nessa abordagem, o PID não utiliza os recursos computacionais como estrutura para seu funcionamento, mas sim controla a utilização dos recursos computacionais com base nos resultados de seu controle.

Couto, Campista e Costa (2014) utilizam uma variação do PID, o PI, como algoritmo de um sistema de controle, o XTC (*Xen Throughput Control*), que monitora o consumo dos recursos de rede de VMs instanciadas no hipervisor Xen, utilizando o resultado da variável de controle para ajustar o *cap*[1] do processador, controlando a vazão de rede para roteadores virtuais baseados em Xen.

Segundo os autores, foi utilizado a variação PI, do algoritmo PID, por ela possibilitar erro

[1] O *cap* impõe um limite rígido na utilização de CPU, indicando a porcentagem máxima do tempo da CPU dada para cada VM. Por exemplo, um domínio com um *cap* de valor 50 poderá utilizar 50% da capacidade de CPU física. Em máquinas com mais de um núcleo de CPU, o valor do *cap* pode chegar até $n * 100\%$, onde n é o número de núcleos disponíveis para um determinado domínio.

zero em estado de equilíbrio em um baixo tempo. A implementação da ação derivada (D) reduziria o tempo de equilíbrio, mas poderia causar oscilações devido ao ambiente de redes de computadores possuir alta variabilidade na saída.

Sobre a utilização da variação PI do PID, Ang, Chong e Li (2005) dizem que 80% das implementações do PID não utilizam a ação derivada (D), justificando que essa ação torna o ambiente instável. Segundo os autores, quando o valor de referência muda ou há ruído no circuito, a derivada pode resultar num sinal de controle teoricamente infinito. Para evitar esse impulso no sinal de controle, a maioria dos pacotes de *software* PID e módulos de hardware adicionam um filtro para o diferencial. A filtragem é particularmente útil em ambientes ruidosos, porém, neste trabalho, também utilizaremos a apenas a variação PI assim como Couto, Campista e Costa (2014).

5.2 Sistema de Controle para Escalabilidade Eficiente de VNF

Esta seção analisa a compatibilidade do controlador PI à arquitetura proposta para escalabilidade eficiente de VNFs. Para isso, são empregados conceitos da teoria de controle de malha fechada para sistemas computacionais(HELLERSTEIN et al., 2004) e (JANERT, 2013).

Inicialmente é apresentado o diagrama de blocos e as funções de transferência de todo sistema; verificada a acurácia do modelo com base nos dados obtidos na seção 4.2; determinado os valores dos parâmetros proporcional e integral do controlador PI e, por fim, realizadas simulações na ferramenta Simulink do MATLAB validando a utilização do controlador PI na escalabilidade automática da VNF NAT em instâncias ClickOS.

Neste trabalho, todo procedimento de cálculo dos parâmetros do modelo, avaliação de correspondência das equações e análise de acurácia utilizam técnicas básicas de teoria de controle, como as também apresentadas em Couto, Campista e Costa (2014).

5.2.1 Diagrama de Blocos

O sistema de controle tratado neste trabalho, Figura 5.2, é baseado no conceito de malha fechada (ÅSTRÖM; HÄGGLUND, 1995). Nele é possível observar que a diferença entre o resultado previsto (y_{sp}) e o resultado obtido do *Processo Xen* (y), é considerado como erro (e) que serve de parâmetro de entrada no *Controlador PI*, cuja saída (u) corresponde à variável que interfere no *Processo Xen* na busca de uma nova saída (y) controlada.

A Figura 5.3 apresenta o diagrama de blocos completo do sistema de controle. Nele, os

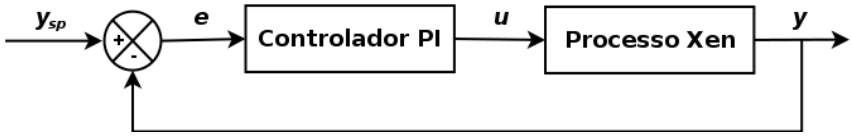


Figura 5.2: Diagrama de Blocos do Sistema de Controle de Malha Fechada.

blocos funcionais estão interligados através de setas com suas funções de transferência representadas por expressões no domínio Z (HELLERSTEIN et al., 2004).

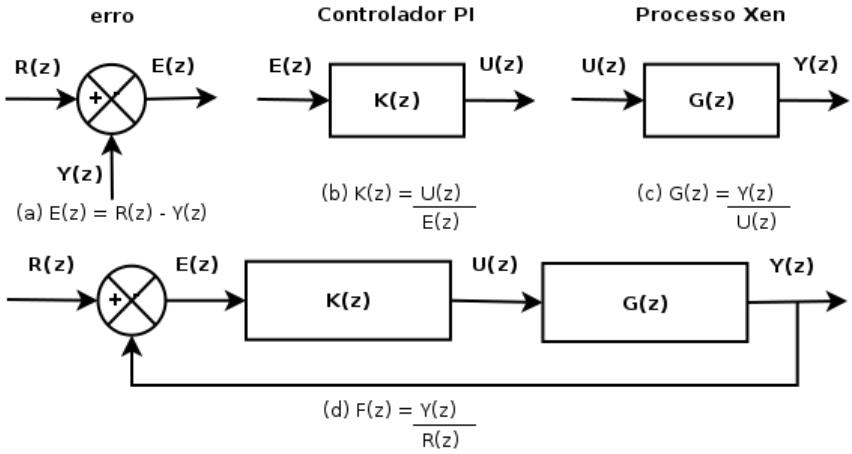


Figura 5.3: Componentes do sistema de controle para escalabilidade automática de recursos NFV. R(z) é a vazão esperada. Y(z) a vazão resultante da VNF. E(z) o erro observado entre R(z) e Y(z). K(z) é a função de transferência do *Controlador PI*, U(z) o sinal de controle gerado por K(z). G(z) a função de transferência do *Processo Xen*. F(z) a função de transferência de todo sistema de controle.

Na Figura 5.3 é possível observar que não foram consideradas variáveis externas ao modelo que pudessem interferir de alguma forma nos resultados, pois, como o sistema é baseado na comparação entre um valor pré-definido e o resultado final de todo processamento, espera-se que ele seja capaz de tratar indiretamente qualquer tipo de interferência.

5.2.2 Processo Xen

A relação entre as entradas de controle e as saídas resultantes de um sistema podem ser mensuradas através de equações diferenciais lineares (HELLERSTEIN et al., 2004) e (JANERT, 2013). Este tipo de equação relaciona saídas atuais e passadas às entradas atuais e passadas. Nessa linha, a equação de primeira ordem, Equação 5.4, é considerada suficiente para modelar o *Processo Xen*, além disso, segue o modelo de única entrada, a variável de controle (u), e única saída, a vazão do sistema (y) (SISO), simplificando sua utilização.

$$y(k+1) = ay(k) + bu(k) \tag{5.4}$$

Na Equação 5.4, *y(k)* corresponde à vazão de saída obtida na VNF e *u(k)* ao *cap* a ser ajustado como entrada do sistema na amostra *k*. Já *a* e *b* são os parâmetros do sistema. A Equação 5.5 resolve a função de transferência do *Processo Xen*, Figura 5.3c, baseado na Equação 5.4.

$$G(z) = \frac{b}{z-a} \tag{5.5}$$

Os parâmetros *a* e *b* foram calculados através do paradigma da caixa-preta utilizando regressão por mínimos quadrados (WANG; ZHU; SINGHAL, 2005). Nesse paradigma o modelo de um processo pode ser definido apenas analisando a relação entre um conjunto de entradas e suas saídas correspondentes. Neste trabalho, tais dados correspondem aos apresentados na Figura 4.6, porém, utilizando o $log(cap)$ no lugar do *cap* puro.

O fluxo de dados utilizado no cálculo da regressão por mínimos quadrados foi o 100Kp/s, para isso, seus valores médios de vazão versus $log(cap)$ (y = 73.6475Kp/s e u = 1.5982) foram considerados ponto de operação, resultando a = 0.2429 e b = 46.6535.

Para validar a acurácia do modelo foram utilizadas a variância de y, R^2, e o gráfico de resíduos. O modelo utilizado resultou R^2 = 0.9901, sugerindo boa acurácia. A Figura 5.4 representa os resultados do gráfico de resíduos. A acurácia do modelo é analisada pela perfeição da sobreposição dos resultados. A reta representa os dados de vazão versus $log(cap)$ previstos pelo modelo, enquanto os asteriscos representam os resultados reais de vazão versus $log(cap)$ observados nos experimentos da seção 4.2.

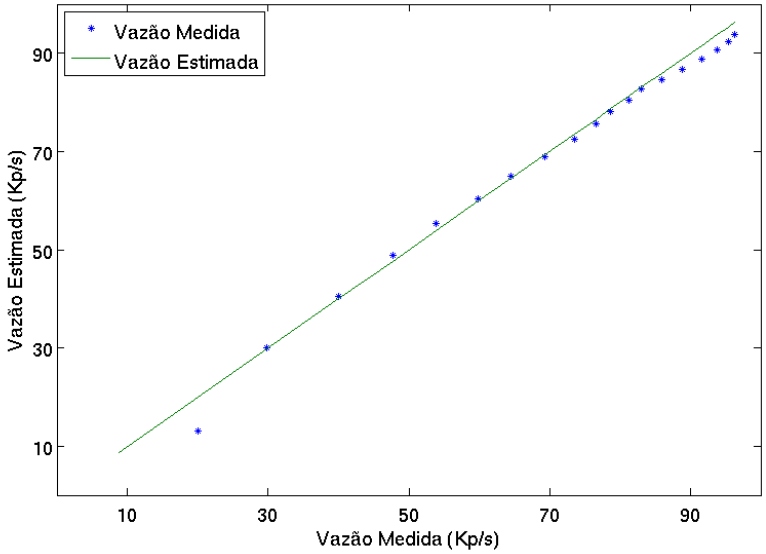


Figura 5.4: Resíduos do modelo.

Como resultado final, é possível observar que a Equação 5.4 modela o *processo Xen* tratado

nesta seção.

5.2.3 Controlador PI

O controlador utilizado é do tipo PI. Ele age sobre o sistema através dos ganhos proporcional (K_p), que determina a taxa de resposta para o sinal de erro, controlando a velocidade da resposta do sistema de controle; e integral (K_i), que soma o termo de erro ao longo do tempo, conduzindo a zero o erro no regime estacionário.

O bloco *Controlador PI* do diagrama de blocos, Figura 5.2, possui a lei de controle dada pela Equação 5.6.

$$u(k) = u(k-1) + (K_p + K_i)e(k) - K_p e(k-1) \tag{5.6}$$

A partir da Equação 5.6, deve-se buscar valores para K_p e K_i que atendam os requisitos de tempo de assentamento (M_s), tempo para alcançar o regime estacionário; e *overshoot* (M_p), vazão máxima atingida acima da vazão em regime estacionário. A Equação 5.7 resolve a função de transferência do *Controlador PI* baseado na Equação 5.6.

$$G(z) = K_p + \frac{K_i z}{z-1} \tag{5.7}$$

Para definição dos valores de K_p e K_i foi utilizado o método de posicionamento dos pólos de forma a atender um M_s e M_p satisfatórios ao modelo, para isso, inicialmente foram estimados M_s = 7 amostras e M_p = 8%. Após algumas manipulações algébricas, os valores retornados foram K_p = -0.0016 e K_i = 0.0099.

Para avaliar se K_p e K_i atendem aos requisitos do projeto, todo sistema de controle foi simulado na ferramenta Simulink do MATLAB. Na simulação foi observado que o sistema alcançou o regime estacionário em 8 amostras, porém, atingiu cerca de 85% da vazão prevista em apenas 3 amostras; além disso, não houve *overshoot*.

Os dados retornados foram considerados satisfatórios para o ambiente proposto, pois, o número de amostras para atingir o regime estacionário ficou apenas uma unidade acima do previsto; e a ausência de *overshoot*, aliada ao fato do sistema ter atingido aproximadamente 85% da vazão prevista em apenas 3 amostras superaram as expectativas.

Capítulo 6

TRABALHOS RELACIONADOS

Apesar da existência de dezenas de provas de conceito (PoC) sobre o NFV (GS, 2014), nenhuma trata especificamente da aplicabilidade das tecnologias existentes atualmente em ambientes de PSI de pequeno e médio porte, mais especificamente, que utilizam como base recursos computacionais com poder de processamento limitado e custo reduzido.

(COUTO; CAMPISTA; COSTA, 2014) apresentou o *Xen Throughput Contol* (XTC), mecanismo que controla automaticamente a escalabilidade de roteadores virtuais baseado no sistema operacional Linux, com capacidade de vazão ajustada através da atuação de um controlador do tipo Proporcional Integral (PI), que ajusta o *cap* do processador do Linux virtualizado no Xen (ONGARO; COX; RIXNER, 2008) de acordo com a necessidade de vazão da instância XTC.

Apesar do controlador de escalabilidade automática do XTC ter apresentado bons resultados, o uso de instâncias Linux como sistema base para VNF não atende aos requisitos de pouco consumo dos recursos de *hardware* do NFV, além disso, a VNF tratada por (COUTO; CAMPISTA; COSTA, 2014) difere da VNF base deste trabalho, gerando um comportamento diferente no ambiente.

A fim de resolver esse problema, definiu-se o ClickOS como sistema base para a VNF tratada neste trabalho. Sobre o ClickOS, (MARTINS et al., 2013) apresentou resultados de testes de desempenho, onde uma instância de apenas poucos MB de memória RAM e espaço em disco atingiu vazão de até 10Gbps com a VNF de roteamento. Neste ambiente, foi utilizado interfaces de rede IXGBE de 10Gbps compatíveis com um mecanismo aprimorado de troca de dados entre a interface de rede e a instância ClickOS. Tal mecanismo atua diretamente no *netfront* e *backend* do Xen, porém, é compatível apenas com interfaces de rede IXGBE e é baseado no switch virtual Vale com suporte a Netmap (RIZZO, 2012). Todas essas características tornaram o ambiente apresentado por (MARTINS et al., 2013) incompatível ao ambiente tratado neste

trabalho.

(OLTEANU; HUICI; RAICIU, 2015) apresentou resultados de um estudo de viabilidade do ClickOS como sistema base para a VNF NAT com manutenção de estado em ambiente computacional distribuído, com um switch Openflow (MCKEOWN et al., 2008) distribuindo a carga entre os servidores. Neste ambiente, atingiu-se vazão de 40Gbps com pacotes de 64Bytes, porém, como em (MARTINS et al., 2013), *hardware* e *software* específicos e dedicados foram utilizados nos experimentos, comprometendo a viabilidade do estudo com o ambiente foco deste trabalho.

Capítulo 7

Conclusão e Trabalhos Futuros

Este trabalho apresentou um modelo de arquitetura responsável por garantir a escalabilidade eficiente de recursos NFV. Para isso, a VNF NAT foi instanciada no sistema minimalista ClickOS em ambiente computacional de baixo custo. A fim de avaliar a viabilidade da arquitetura, foram confrontados a vazão máxima obtida, consumo de recursos de *hardware* e suporte à escalabilidade automática da VNF com os resultados observados em instâncias baseadas em Linux virtualizado. Nesse ambiente, o ClickOS apresentou vazão de rede superior em até 900% ao Linux virtualizado, consumo de recursos de *hardware* cerca de 10 vezes menor, e estrutura e comportamento compatíveis à automação da escalabilidade da VNF através de um controlador PI. A análise final dos resultados indicaram que o ClickOS atende aos requisitos de alto desempenho, escalabilidade e baixo consumo de recursos de *hardware* do hospedeiro, requisitos inerentes ao NFV, viabilizando sua implantação em ambientes com as características definidas neste trabalho.

Como trabalhos futuros, pretende-se utilizar a porção derivativa (D) do controlador PID, além de realizar novos estudos com controladores que sigam outras lógicas ou algoritmos, confrontando os resultados aqui apresentados. Também pretende-se avaliar a viabilidade da utilização de outros sistemas minimalistas como base para VNFs.

REFERÊNCIAS

ANG, K. H.; CHONG, G.; LI, Y. Pid control system analysis, design, and technology. *Control Systems Technology, IEEE Transactions on*, v. 13, n. 4, p. 559–576, July 2005. ISSN 1063-6536.

BIFULCO, R. et al. Rethinking access networks with high performance virtual software brases. In: *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. [S.l.: s.n.], 2013. p. 7–12.

BRISCOE, B. *Network Functions Virtualisation*. 2013. Disponível em: ⟨http://www.ietf.org/proceedings/86/slides/slides-86-sdnrg-1.pdf⟩.

COUTO, R. S.; CAMPISTA, M. E. M.; COSTA, L. H. M. n. "Network Resource Control for Xen-based Virtualized Software Routers", in Computer Networks, Elsevier, ISSN 1389-1286, DOI. 10.1016/j.comnet.2014.02.003, vol. 64, p. 71–88, May 2014.

CSáSZáR, A. et al. Unifying cloud and carrier network: Eu fp7 project unify. In: *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Washington, DC, USA: IEEE Computer Society, 2013. (UCC '13), p. 452–457. ISBN 978-0-7695-5152-4. Disponível em: ⟨http://dx.doi.org/10.1109/UCC.2013.89⟩.

ETSI. *Network Functions Virtualization (NFV): Architectural Framework*. [S.l.]: ETSI, 2013. (GS NFV 002 V1.1.1, 2013).

FAYAZBAKHSH, S. K.; REITER, M. K.; SEKAR, V. Verifiable network function outsourcing: Requirements, challenges, and roadmap. In: *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. New York, NY, USA: ACM, 2013. (HotMiddlebox '13), p. 25–30. ISBN 978-1-4503-2574-5. Disponível em: ⟨http://doi.acm.org/10.1145/2535828.2535831⟩.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn. *Queue*, ACM, New York, NY, USA, v. 11, n. 12, p. 20:20–20:40, dez. 2013. ISSN 1542-7730. Disponível em: ⟨http://doi.acm.org/10.1145/2559899.2560327⟩.

GOODWIN, G. C.; GRAEBE, S. F.; SALGADO, M. E. *Control System Design*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN 0139586539.

GS. ETSI GS NFV 001 V1.1.1: Network Functions Virtualisation (NFV); Use Cases. ⟨http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf⟩. 2013.

GS. ETSI GS NFV 002 V1.1.1: Network Functions Virtualisation (NFV); Architectural Framework. ⟨http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf⟩. 2013.

GS. ETSI GS NFV 004 V1.1.1: Network Functions Virtualisation (NFV); Virtualisation Requirements. ⟨http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf⟩. 2013.

GS. ETSI NFV Proofs of Concept. ⟨http://www.etsi.org/technologies-clusters/technologies/nfv/nfv-poc⟩. 2014.

GUPTA, R. A.; CHOW, M.-Y. Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics*, v. 57, n. 7, p. 2527–2535, 2010. Disponível em: ⟨http://dblp.uni-trier.de/db/journals/tie/tie57.html\#GuptaC10⟩.

HELLERSTEIN, J. L. et al. *Feedback Control of Computing Systems*. [S.l.]: John Wiley & Sons, 2004. ISBN 047126637X.

JANERT, P. K. *Feedback Control for Computer Systems*. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449361692, 9781449361693.

JIN, X. et al. Softcell: Scalable and flexible cellular core network architecture. In: *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*. New York, NY, USA: ACM, 2013. (CoNEXT '13), p. 163–174. ISBN 978-1-4503-2101-3. Disponível em: ⟨http://doi.acm.org/10.1145/2535372.2535377⟩.

JOHN, W. et al. Research directions in network service chaining. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–7.

KIM, H.; FEAMSTER, N. Improving network management with software defined networking. *Communications Magazine, IEEE*, v. 51, n. 2, p. 114–119, February 2013. ISSN 0163-6804.

KOBAYASHI, M. et al. Maturing of openflow and software-defined networking through deployments. *Computer Networks*, n. 0, p. –, 2013. ISSN 1389-1286. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S138912861300371X⟩.

KOHLER, E. et al. The click modular router. *ACM Trans. Comput. Syst.*, ACM, New York, NY, USA, v. 18, n. 3, p. 263–297, ago. 2000. ISSN 0734-2071. Disponível em: ⟨http://doi.acm.org/10.1145/354871.354874⟩.

MANZALINI, A.; SARACCO, R. Software networks at the edge: A shift of paradigm. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–6.

MARTINS, J. et al. Enabling fast, dynamic network processing with clickos. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 67–72. ISBN 978-1-4503-2178-5. Disponível em: ⟨http://doi.acm.org/10.1145/2491185.2491195⟩.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, p. 69–74, March 2008. ISSN 0146-4833.

MENDEL, J. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, v. 83, n. 3, p. 345–377, Mar 1995. ISSN 0018-9219.

NFV White Paper. Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & C all for Action. Issue 1. ⟨http://portal.etsi.org/NFV/NFV_White_Paper.pdf⟩. 2012.

OGATA, K. *Engenharia de controle moderno*. [S.l.]: Pearson Prentice Hall, 2003. ISBN 9788587918239.

OLTEANU, V. A.; HUICI, F.; RAICIU, C. Lost in network address translation: Lessons from scaling the world's simplest middlebox. In: *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. New York, NY, USA: ACM, 2015. (HotMiddlebox '15), p. 19–24. ISBN 978-1-4503-3540-9. Disponível em: ⟨http://doi.acm.org/10.1145/2785989.2785994⟩.

ONF. OpenFlow-enabled SDN and Network Functions Virtualization. ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf⟩. 2014.

ONGARO, D.; COX, A. L.; RIXNER, S. Scheduling i/o in virtual machine monitors. In: *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. New York, NY, USA: ACM, 2008. (VEE '08), p. 1–10. ISBN 978-1-59593-796-4. Disponível em: ⟨http://doi.acm.org/10.1145/1346256.1346258⟩.

PENTIKOUSIS, K.; WANG, Y.; HU, W. Mobileflow: Toward software-defined mobile networks. *Communications Magazine, IEEE*, v. 51, n. 7, p. 44–53, July 2013. ISSN 0163-6804.

PONGRáCZ, G. et al. Cheap silicon: A myth or reality? picking the right data plane hardware for software defined networking. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 103–108. ISBN 978-1-4503-2178-5. Disponível em: ⟨http://doi.acm.org/10.1145/2491185.2491204⟩.

QUINN, E. P.; NADEAU, E. T. *Service Function Chaining Problem Statement*. [S.l.], 2013.

RISSO, F.; MANZALINI, A.; NEMIROVSKY, M. Some controversial opinions on software-defined data plane services. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–7.

RIZZO, L. NETMAP: A Novel Framework for Fast Packet I/O. Usenix ATC'12 (Best paper award), Boston, USA, Junho, 2012. 2012.

ROSA, R. V. et al. Network function virtualization: Perspectivas, realidades e desafios. In: *Minicursos do Simpósio Brasileiro de Redes de Computadores (SBRC 2014)*. [S.l.: s.n.], 2014.

SAMDANIS, K. et al. Virtual bearer management for efficient mtc radio and backhaul sharing in lte networks. In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*. [S.l.: s.n.], 2013. p. 2780–2785. ISSN 2166-9570.

SCOTT-HAYWARD, S.; O'CALLAGHAN, G.; SEZER, S. Sdn security: A survey. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–7.

SELLERS, D. An overview of proportional plus integral plus derivative control and suggestions for its successful application and implementation. In: *Proceedings for the 2001 International Conference on Enhanced Building Operations*. [S.l.: s.n.], 2001.

SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. *Communications Magazine, IEEE*, v. 51, n. 7, p. 36–43, July 2013. ISSN 0163-6804.

SIRACUSA, D.; SALVADORI, E.; RASHEED, T. Edge-to-edge virtualization and orchestration in heterogeneous transport networks. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–6.

TRILOGY2. *EU FP7 Trilogy2 project: Building the Liquid Net*. 2014. ⟨http://www.trilogy2.org/⟩.

WANG, Z.; ZHU, X.; SINGHAL, S. Utilization and slo-based control for dynamic sizing of resource partitions. In: *Ambient Networks*. [S.l.]: Springer, 2005. p. 133–144.

ÅSTRÖM, K. J.; HÄGGLUND, T. *PID Controllers: Theory, Design, and Tuning*. 2. ed. [S.l.]: Instrument Society of America, Research Triangle Park, NC, 1995.