Ativando Funções de Rede Virtualizadas: Modelo de Desenvolvimento e Nova Plataforma

Leonardo da C. Marcuzzo[1], **Vinícius F. Garcia**[1], **Carlos R. P. dos Santos**[1]

[1]Universidade Federal de Santa Maria (UFSM)
Programa de Pós Graduação em Ciência da Computação (PPGCC)

{lmarcuzzo, vfulber, csantos}@inf.ufsm.br

Abstract. *Network Functions Virtualization (NFV) is an innovative paradigm which aims to use virtualization techniques to replace network functions deployed in specialized hardware. Currently, there is no unanimity of the scientific community in the choice of a platform for the execution of network virtualized functions due to the different implementation models and the lack of completeness in meeting the expected requirements for the NFV paradigm. In this way, this article proposes to detect a standard model for the implementation of platforms capable of executing and managing virtualized network functions, as well as presenting a prototype that satisfies this model, called Click-on-OSv*[1]*, and evaluate this through comparative experiments with other available platforms.*

Resumo. *A Virtualização de Funções de Rede é um paradigma inovador que objetiva utilizar técnicas de virtualização para substituir funções de rede implementadas em hardware especializado. Atualmente não existe unanimidade da comunidade científica na escolha de uma plataforma para execução de funções virtualizadas de rede devido aos diferentes modelos de implementação e da não completude no atendimento a requisitos previstos para o paradigma NFV. Desta forma, este artigo se propõem a detectar um modelo padrão para a implementação de plataformas capazes de executar e gerenciar funções virtualizadas de rede, assim como apresentar um protótipo que satisfaz esse modelo, chamado Click-on-OSv*[2]*, e avaliar esta através de experimentos comparativos com demais plataformas disponíveis.*

1. Introdução

A atual infraestrutura de rede, baseada na utilização de equipamentos dedicados (chamados de *middleboxes*) para o processamento de pacotes se tornou a base fundamental para a distribuição e popularização das redes de computadores através do mundo. Entretanto, este paradigma apresenta deficiências em sua arquitetura referentes a vulnerabilidades e limitações no suporte a protocolos. Embora existam propostas visando reduzir os efeitos de tais dificuldades, estas não são completamente efetivas e acabam tornando a Internet como um todo cada vez mais complexa [Taylor and Turner 2004]. Além disso, do ponto de vista protocolar, apesar da possibilidade de realizações de extensões e adaptações do protocolo IP, estas resultam em maiores custos de processamento em cada pacote que o

[1]Available in https://github.com/lmarcuzzo/click-on-osv
[2]Disponível em https://github.com/lmarcuzzo/click-on-osv

implementa ou até mesmo não são suportados pelos equipamentos de rede, impossibilitando seu processamento. Devido a estes motivos, é possível afirmar que a Internet sofre um processo de ossificação, tendo o núcleo da rede sem alterações significativas desde 1993 [Handley 2006].

O paradigma de Virtualização de Funções de Rede (*Network Functions Virtualization* - NFV) utiliza tecnologias de virtualização já existentes para desacoplar a função (ou serviço) de rede de um plano de hardware para um plano de software [ETSI 2012]. Para que essa mudança de paradigma nas telecomunicações seja adotada na prática, trabalhos de padronização, arquitetura e construção de *enablers* (*i.e.*, tecnologias e soluções que possibilitam a implementação e execução do paradigma) devem ser desenvolvidos através de pesquisas envolvendo cooperações entre academia e indústria. Com esse propósito, organizações de grande relevância como a *Internet Engineering Task Force* (IETF) e *European Telecommunications Standards Institute* (ETSI) lideram esforços através de diferentes grupos de trabalho (*e.g.*, IRTF NFV *Research Group* (NFVRG), IETF *Service Function Chaining* (SFC), ETSI NFV Industry Specification Groups (ISG)) discutindo e fundamentando NFV em seus mais diversos aspectos, sendo seus resultados considerados como diretivas que permeiam demais pesquisas na área.

Dentre as necessidades do paradigma NFV estão plataformas capazes de efetivamente executar processamento de pacotes, sendo capazes de aplicar funções de rede virtualizadas observando requisitos de portabilidade, alto desempenho e simplicidade [ISG 2013]. Entre as plataformas disponíveis atualmente destacam-se o ClickOS [Martins et al. 2014] e OpenNetVM [Zhang et al. 2016], estas possibilitam a criação de múltiplas funções de rede e consomem poucos recursos computacionais, entretanto também apresentam limitações relacionadas aos requisitos definidos pela especificação de NFV, mostradas na seção 2. Sendo assim, não existe unanimidade da comunidade científica na escolha por uma plataforma ou outra, sendo necessário avaliar a adequabilidade das mesmas em comparação ao cenário de execução.

Uma vez que as principais plataformas disponíveis não atendem completamente os requisitos necessários em suas implementações, este artigo busca esclarecer práticas e decisões de projeto a serem tomadas de forma a determinar um modelo para implementação de plataformas compatíveis com requisitos de virtualização definidos para NFV. Além disso, apresenta-se uma plataforma para a execução de funções de rede virtualizadas implementada considerando o modelo previamente apresentado, chamado Click-on-OSv, sendo uma solução completa focada na facilidade de implementação, escala e gerência pelos operadores de rede. Finalmente, a plataforma Click-on-OSv é avaliada em diferentes cenários e comparada ao seu concorrente mais próximo.

A seguir, a seção 2 apresenta uma fundamentação teórica sobre o tema de NFV, abordando a sua arquitetura de alto nível e requisitos, bem como trabalhos relacionados ao teor deste artigo e suas limitações que motivam o desenvolvimento de um modelo para criação e uma nova plataforma. Na Seção 3 são apontadas práticas e decisões para implementação de plataformas de execução de funções de rede virtualizadas focando nos requisitos de portabilidade, alto desempenho e simplicidade, sumarizados em um modelo arquitetural. Na Seção 4 a plataforma Click-on-OSv é apresentada assim como as tecnologias utilizadas para sua implementação e testes de desempenho. Finalmente, a Seção 5 conclui o artigo apresentando as expectativas dos autores e futuras inovações previstas

para a plataforma.

2. Fundamentação

Esta seção apresenta uma visão geral da arquitetura de NFV, bem como os requisitos que precisam ser atingidos para uma utilização efetiva do ambiente. Trabalhos relacionados a implementação de plataformas para execução de funções de rede virtualizadas também são apresentados realçando seus pontos positivos e limitações encontradas.

2.1. Arquitetura NFV e Requisitos

O objetivo geral de NFV é utilizar técnicas de virtualização já existentes no mercado para consolidar os mais variados tipos de equipamentos de rede em servidores de virtualização, computadores e sistemas de armazenamento [ETSI 2012]. Isto se faz necessário pois atualmente as redes de grande porte fazem ampla utilização de *middleboxes* que, embora sua utilização resulte em benefícios com relação a segurança, desempenho e redução de uso de banda (*e.g.*, aceleradores WAN), apresentam desvantagens como alto custo para implementação de novas funcionalidades, complexidade de gerenciamento e dificuldade de escalabilidade.

Considerando os problemas que surgem com a utilização de *middleboxes*, estabeleceu-se em 2012 o Grupo de Padrões da Indústria para Virtualização de Funções de Rede (*Network Functions Virtualization Industry Standards Group* - ISG NFV), responsável por publicar as especificações de NFV, como a sua arquitetura, métricas para qualidade de serviço, casos de uso, *framework* de gerência e orquestração e requisitos para sua implantação.

Assim, a arquitetura de NFV é dividida em três blocos distintos, sendo eles a Infraestrutura NFV (*NFV Infrastructure* - NFVI), o *framework* de Gerência e Orquestração (*NFV Management and Orchestration* - NFV MANO) e as Funções Virtualizadas de Rede (*Virtual Network Functions* - VNF). Como pode ser visto na Figura 1, a NFVI fornece os recursos físicos e virtuais que são utilizados pelas VNFs, que por sua vez são responsáveis por executar as funções virtualizadas, enquanto que o *framework* NFV MANO deve realizar orquestração e gerenciar tanto a infraestrutura quanto as funções virtualizadas. Finalmente, e como objetivo deste trabalho, estão as as VNFs capazes de aplicar as funções de rede ao tráfego direcionado a ela e encaminha-lo a um próximo ponto. Junto a cada VNF existe também o Sistema de Gerenciamento de Elemento (*Element Management System* - EMS) que prove funcionalidades relacionadas ao controle do ciclo de vida (*e.g.*, iniciar função, parar função, modificar função) de uma VNF para o orquestrador ou para um operador de rede.

De maneira complementar as definições arquiteturais, em [ISG 2013] são definidos os requisitos de alto nível que devem ser considerados para a criação de plataformas que suportam a execução de VNFs. Cada um destes requisitos é responsável por um aspecto diferente do funcionamento da VNF e visam garantir que as mesmas sejam compatíveis com as características desejáveis ao paradigma NFV:

- **Portabilidade:** para que a execução de VNFs de diferentes empresas em uma variedade de plataformas padronizadas seja possível, é necessário definir uma interface unificada que claramente separa as instâncias das VNFs de sua infraestrutura física, o que pode ser atingido ao utilizar-se virtualização;

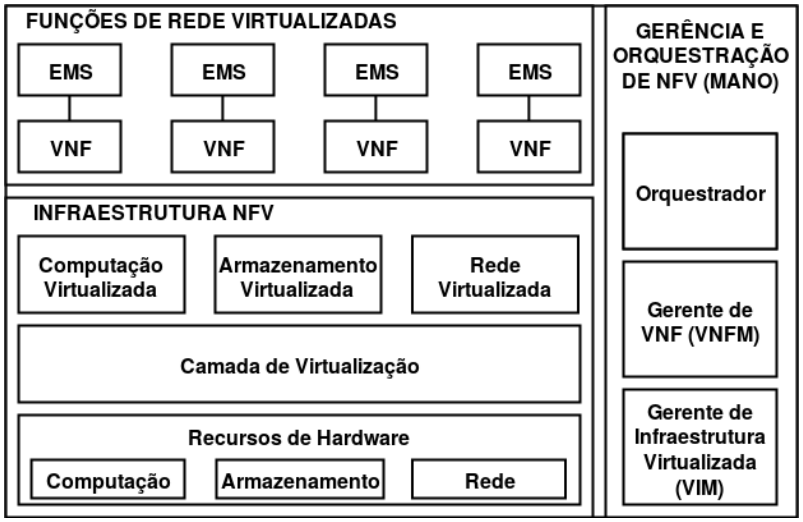

Figura 1. Arquitetura NFV de alto nível

- **Desempenho:** por NFV ser implantada em equipamentos e tecnologias padronizadas (*i.e.,* sem otimizações encontradas em hardware proprietário), é possível que haja uma perda de desempenho com relação a *middleboxes*. O desafio deve ser mitigar a degradação de desempenho o máximo possível, utilizando para isto tecnologias e *hypervisors* apropriados;
- **Integração:** operadores de rede devem ser capaz de utilizarem em conjunto servidores, *hypervisors* e funções virtualizadas de rede de diferentes fornecedores, de modo que isto não incorra custos de desempenho. O ecossistema deve oferecer serviços de integração e configuração, evitando a dependência de um fornecedor específico;
- **Gerência e Orquestração:** um levantamento realizado por [Bondan et al. 2014] apresenta requisitos relacionados a gerência de VNFs do ClickOS, estes podem ser estendidos a outros tipos de plataforma. Neste caso, foram identificados requisitos com relação à configuração do servidor onde as VNFs são executadas, implantação da infraestrutura e a instanciação e monitoramento das VNFs.
- **Escalabilidade:** deve-se garantir que VNFs possam ser facilmente escaladas conforme a necessidade dos operadores de rede. Para isso, utilização de recursos por parte das instâncias deve ser reduzida quando comparada a sistemas tradicionais.

A IETF também demonstra interesse no paradigma NFV através do mapeamento de lacunas de pesquisa, este realizado pelo NFV *Research Group* (NFV-RG) [Bernardos et al. 2018]. A composição de serviços de rede é apresentado como uma dessas lacunas, arquiteturalmente define-se serviço de rede como um conjunto ordenado de funções de rede e o encaminhamento de tráfego através das mesmas - nomeado de Service Function Chaining (SFC) [Halpern and Pignataro 2015]. Para que um serviço de rede seja caracterizado como SFC é necessário que ao menos um elemento de processamento de tráfego - Service Function (SF) - apresente as características esperadas do paradigma NFV, em outras palavras, seja uma VNF. Porém, para a implantação eficiente de SFCs, as VNFs envolvidas devem suportar o processamento e atualização de metadados vinculados aos pacotes - dispensando assim a utilização de *proxies*. Esse grupo de metadados de SFCs são apresentados pela IETF como o Cabeçalho de Serviços de Rede

(*Network Service Header* - NSH) [Quinn et al. 2018]. O NSH encapsula os pacotes com informações de encaminhamento e controle do tráfego entre os possíveis caminhos dentro de uma mesma SFC. Sendo assim, espera-se que *frameworks* destinados a implementação de VNFs sejam capazes de processar o NSH de maneira nativa, provendo funções de alto nível com o intuíto de facilitar a remoção, atualização e reinserção de tal cabeçalho.

2.2. Trabalhos Relacionados

Trabalhos abordando o paradigma NFV são abundantes na literatura demonstrando a atenção que a comunidade científica dedica ao tema. Algumas iniciativas buscam oferecer plataformas capazes de executar funções de rede virtualizadas, nesta subseção objetiva-se apresentar as principais propostas já consolidadas.

O ClickOS [Martins et al. 2014] é uma plataforma otimizada para a execução de VNFs através de virtualização, baseando-se no *unikernel* Mini-OS, acelerador de pacotes *netmap* e no *Click Modular Router*, onde as funções de rede são executadas. Sua arquitetura tem como foco o desempenho, flexibilidade, isolamento de ambiente e escalabilidade. Ao utilizar técnicas como a paravirtualização e compartilhamento de uma área de memória entre a VM e o *hypervisor*, em conjunto com modificações realizadas no sistema de entrada e saída do *hypervisor*, além de um comutador virtualizado específico (VALE), o ClickOS é capaz de saturar *links* de 10GbE, utilizando um único núcleo de processamento. Embora atenda os requisitos de desempenho e escalabilidade, o ClickOS é capaz de executar apenas em um *hypervisor* (Xen), quebrando o requisito de portabilidade, além de não oferecer uma interface de gerência nativa, uma vez que sua gerência pode ser realizada apenas localmente, através da solução XenStore.

O OpenNetVM [Zhang et al. 2016] apresenta, por sua vez, uma plataforma flexível e simplificada que utiliza contêineres (*e.g.*, LXC) para a execução de VNFs em servidores em conjunto com o *framework* de aceleração de pacotes DPDK [Intel 2014]. Ao utilizar contêineres, a sobrecarga criada por uma camada de virtualização é eliminada, de forma que as instâncias das VNFs são executadas diretamente no servidor utilizando o próprio sistema operacional, mas de forma isolada. O OpenNetVM também atende os requisitos de portabilidade e escalabilidade uma vez que contêineres resultam em menores sobrecargas em relação a máquinas virtuais e podem ser instanciados em grande quantidade. Os requisitos de gerenciamento também são atendidos, pois sua arquitetura já inclui o *NF Manager*, que pode ser utilizado para gerenciar as instâncias sendo executadas em um servidor. No entanto, a utilização de contêineres ocasiona problemas com relação a portabilidade e a segurança, pois a migração dos mesmos pode ser feita apenas para uma infraestrutura semelhante, com o mesmo sistema operacional e *framework* de aceleração de pacotes. Além disso, dispositivos de hardware (*e.g.*, interfaces de rede) não podem ser compartilhadas entre as instâncias, limitação não existente em máquinas virtuais paravirtualizadas.

3. Modelo de Desenvolvimento

Tecnologias e ferramentas que contribuem de alguma forma para o desenvolvimento e implantação de NFV são chamados de NFV *Enablers*. Dentre esses, são exemplos os servidores de virtualização, *hypervisors*, comutadores virtuais, *frameworks* para aceleração de pacotes e para criação e implantação das funções virtualizadas de rede, APIs para

automação de gerência e instanciação de máquinas virtuais e tecnologias para gerência do plano de dados de rede (*e.g.*, *OpenFlow*) [ETSI 2012]. Uma vez que esses NFV *Enablers* tipicamente atendem apenas parte dos requisitos para a implantação de NFV, é possível utilizá-las em conjunto, de modo que estes requisitos sejam atendidos em completude.

A categorização desses NFV *Enblers* é um processo fundamental para que um modelo de desenvolvimento de plataformas destinadas a execução de funções de rede virtualizadas seja criado. Sendo assim, decisões de projeto e a escolha de ferramentas já consolidadas podem ser realizadas baseada em um mapeamento de características desejáveis para satisfazer os requisitos de portabilidade, desempenho, integração, gerência e escalabilidade. Nesse sentido, quatro categorias de NFV *Enablers* foram identificadas: sistemas operacionais, *frameworks* para aceleração de pacotes, *frameworks* para criação e execução de VNFs e sistemas de gerenciamento.

Em relação aos sistemas operacionais, aqueles considerados tradicionais são projetados para serem robustos, utilizados por múltiplos usuários com o intuito de realizar várias tarefas ao mesmo tempo com os mais diversos objetivos, resultando em uma utilização de recursos com funcionalidades que não são necessárias no contexto de NFV. Em contêineres, o sistema hospedeiro aloca uma quantia de recursos para um certo processo ou função e o isola do resto do sistema operacional contêineres. Assim, a função ainda pode executar em conjunto com o sistema hospedeiro mas utilizando apenas os recursos que foram disponibilizados a mesma. No entanto, a utilização de contêineres não satisfaz o requisito de integração, pois a função virtualizada deve suportar o sistema operacional e o *hardware* associado, além de também não satisfazer o requisito de portabilidade, uma vez que para mover um contêiner de um servidor de virtualização a outro, é necessário que estes possuam um *hardware* e sistema operacional compatível.

O conceito de sistemas operacionais *unikernel* apresenta um paradigma diferente quando comparado as demais opções, mantendo o foco na simplicidade, escalabilidade, segurança e desempenho. Em um *unikernel*, o núcleo é compilado em uma biblioteca conectada diretamente ao binário da aplicação, com o único propósito de prover as funções necessárias para a execução do aplicativo e sua comunicação com o *hypervisor*, criando assim uma máquina virtual especializada, isolada e segura. Devido a isso, a utilização de sistemas *unikernel* atende ao requisito de portabilidade e, apesar de em geral necessitarem de maior quantidade de recursos em relação a contêineres, ainda apresentam necessidades muito reduzidas quando comparados a sistemas tradicionais, satisfazendo o requisito de escalabilidade. Finalmente, questões relacionadas a integração devem ser observadas a fim de evitar que dependências a sistemas secundários ou plataformas de virtualização específicas restrinjam o *unikernel* escolhido.

Apesar de *unikernels* serem otimizados para execução em ambientes virtualizados, sua pilha de rede, assim como a de sistemas tradicionais, tipicamente não apresenta o desempenho necessário para usufruir totalmente de interfaces de rede modernas (*e.g.*, 10GbE, 40GbE e 100GbE) e mesmo quando suportam tais interfaces, estes requerem modificações extensivas na arquitetura das aplicações. Devido a isto, *frameworks* para a aceleração de pacotes têm sido apresentados como uma alternativa para a atualização desses sistemas. A função destes *frameworks*, em suma, é realizar a substituição da pilha de rede focando na otimização dos recursos utilizados para o encaminhamento e processamento de pacotes. Dentre as técnicas utilizadas por estes *frameworks* para atingir um

alto desempenho estão a pré-alocação de *buffers* de memória para os pacotes, o uso de *polling* (*i.e.*, o sistema verifica a interface e pré-aloca na memória os pacotes ao invés de esperar uma interrupção) e o processamento dos pacotes em lotes. Ao utilizar-se destas ferramentas, os requisitos de alto desempenho e de escalabilidade de rede são atendidos.

Ferramentas que efetivamente são responsáveis por criar e executar as funções virtualizadas de rede são consideradas como *frameworks* para criação e execução de VNFs. Mesmo que os *frameworks* de aceleração de pacotes sejam efetivos para realizar o encaminhamento de pacotes e outras funções simples, eles não são adequados para a implementação de funções e serviços de rede mais complexos (*e.g.*, IDS, *statefull firewalls* e outras funções de roteamento avançado). Assim, é necessário a utilização de ferramentas desenvolvidas especialmente para a criação e execução destas funções mais avançadas. Espera-se que esses *frameworks* provenham interfaces de programação de alto nível, já dispondo de um largo conjunto de funções especializadas no tratamento de pacotes e comunicação com interfaces de rede. Além disso, a extensibilidade do mesmo é um fator determinante, dando liberdade a quem implementa a plataforma para execução de funções de rede virtualizadas de adaptar o *framework* escolhido se necessário.

Sistemas de gerenciamento providenciam formas de gerenciar os componentes utilizados para a implantação da plataforma de maneira remota e são uma necessidade para flexibilizar o processo de gerenciamento da rede. Funções relacionadas ao ciclo de vida básico da plataforma, como ligar, desligar e atualizar, necessitam de uma interface externa acessível, seja para ser usado pelo MANO quando presente (mais especificadamente pelo VNFM) ou seja para ser utilizado diretamente pelo operador de rede. Ademais, métricas de monitoramento (*e.g.*, uso de CPU, uso de memória, tráfego recebido, tráfego encaminhado) também devem ser disponibilizados nessa interface. Finalmente, o sistema de gerenciamento de um VNF deve ser facilmente portado junto a mesma e genérica o suficiente para ser compatível com diferentes sistemas hospedeiros.

Do ponto de vista arquitetural, uma VNF pode ser composta de um ou mais componentes, chamados de *Virtual Network Function Components* (VNFC) [ETSI 2012]. Cada um destes componentes é responsável por uma funcionalidade específica e podem ser materializados como diferentes NFV *Enablers* operando junto para executar a plataforma como um todo. Considerando a categorização apresentada como exemplares de VNFCs, a Figura 2 apresenta um modelo de desenvolvimento genérico para a implementação de plataformas para executar funções de rede virtualizadas.

Esta arquitetura genérica também apresenta um componente inédito, chamado agente estrutural, cuja responsabilidade é interconectar os demais subsistemas que compõem a arquitetura. De modo simplificado, o agente estrutural funciona como um pré-processador de pacotes, enviando os pacotes que entram na VNF para os respectivos sistemas que devem processá-los. Por exemplo, dado que exista suporte para diferentes *frameworks* para criação de funções virtualizadas, que por sua vez podem ser implementados em linguagens diferentes, a função do agente estrutural é fornecer uma interface de entrada em comum - semelhante a ideia do ScriptMIB [Levi and Schönwälder 2001] - para estes diferentes *frameworks*, criando uma abstração para os demais componentes da rede que enviam e recebem tráfego da VNF. O agente estrutural também é responsável pelo processamento do NSH, que definem, entre outras informações, se os pacotes são efetivamente direcionados a VNF ou, se a VNF faz parte de uma SFC, realizam o con-

trole da posição dos pacotes na cadeia de serviço, baseando-se para isso em um classificar SNMP v3 [Blumenthal and Wijnen 2002] com suporte a versões legadas.

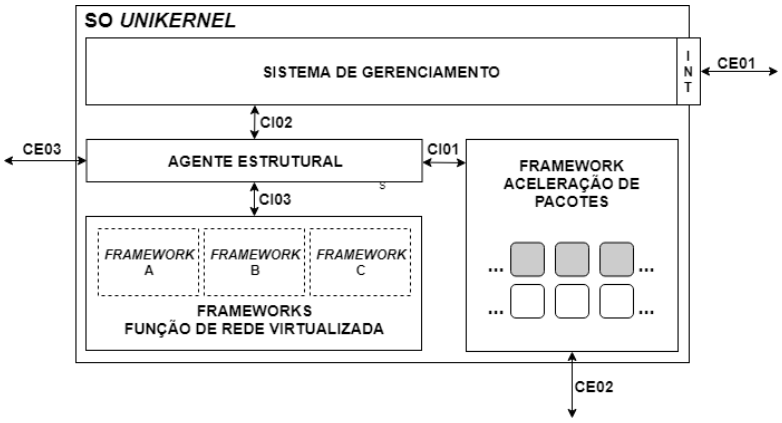

Figura 2. Modelo de Desenvolvimento para Plataformas

O sistema de gerenciamento, por sua vez, é responsável por receber requisições para reconfiguração ou monitoramento dos componentes internos da VNF - atuando como um EMS. Através do ponto de conexão CE01, um operador de rede, ou então um sistema de orquestração de NFV (através do VNFM), é capaz de se comunicar diretamente com cada uma das VNFs que compõem a infraestrutura, podendo então realizar requisições através de um protocolo (*e.g.*, REST, SOAP) para reconfigurar a VNF de acordo com as necessidades do mesmo. Este mesmo ponto de conexão também pode ser utilizado pela infraestrutura NFV ou por um operador para coletar estatísticas de uso de recursos, pacotes processados e regras de configuração instaladas, permitindo o monitoramento para propósitos de migração ou escalabilidade. Internamente, o sistema de gerenciamento comunica-se com o agente estrutural pelo ponto de conexão CI02, já que configurações do NSH e definição dos *frameworks* devem ser processadas inicialmente por este agente, que em um segundo delega regras específicas aos outros componentes da VNF.

Já os *framework* de aceleração de pacotes podem ser escolhidos de acordo com a necessidade do usuário. Exemplos incluem Intel DPDK [Intel 2014], que quando associado com placas Intel, é capaz de saturar *links* de 40GbE mesmo que executando em máquinas virtuais, e *netmap* [Rizzo 2012], que possui suporte a diversas placas de rede. A função destes *frameworks* dentro da arquitetura é aumentar o desempenho das VNFs, dado que a pilha de rede do sistema operacional não é otimizado para grandes quantidades de pacotes. Através do ponto de conexão CE02, estes *frameworks* recebem e enviam pacotes para a infraestrutura, podendo estes pontos estarem conectados tanto a interfaces de rede físicas como virtualizadas. O ponto de conexão CI01 é utilizado para a configuração do *framework* pelo agente estrutural, e também para o envio e recebimento de pacotes de outros componentes para o ponto de saída. Por outro lado, determinadas circunstâncias de execução podem exigir economia de recursos, e o uso dos *frameworks* de aceleração de pacotes se torna inviável - nesse caso o próprio agente estrutural é configurado para acessar diretamente as interfaces de rede através da conexão CE03, é importante ressaltar que a utilização da aceleração de pacotes ou do acesso direto é uma opção exclusiva, em outras palavras, quando um é utilizado o outro é inativado afim de manter a coerência interna do encaminhamento de pacotes.

Por fim, os *frameworks* de funções de rede virtualizadas são os responsáveis por realizar as funções de processamento e manipulação de pacotes da VNF. Estes *frameworks* podem, dentre outras funções, processar, modificar e marcar pacotes em camadas 2 e 3. Considerando a flexibilidade, a implementação destes pacotes é agnóstica ao restante da arquitetura, podendo ser compostos de vários componentes e implementados em linguagens diferentes, dado que os pontos de comunicação destes *frameworks* mantenham a compatibilidade com os outros componentes. O ponto de conexão CI03 é por onde o agente estrutural configura, recupera estatísticas, envia pacotes para serem processados e recebem o posterior resultado desse processamento.

No modelo proposto, cada componente possui um papel único na arquitetura e são interligados através de um agente estrutural utilizando um conjunto de conexões padronizadas, sendo assim todos os componentes apresentam execução independente dentro de um mesmo sistema operacional *unikernel*. Isto resulta em uma arquitetura modular, onde a substituição de um componente ocorre de maneira natural, ou seja sem necessidade modificações colaterais dentro da plataforma. Desta forma, VNFs podem ser compostas utilizando os componentes mais adequados para cada situação. Por exemplo, considerando a implementação de uma SFC, ao invés de cada VNF que a compõem possuir um *framework* completo para desenvolvimento de funções de rede, considera-se implementar apenas um VNFC específico capaz de realizar a função destinada a cada ponto da cadeia de serviço, reduzindo assim a necessidade de recursos para a instanciação da mesma. Cenários com limitações de recursos (*e.g.*, *Fog Computing*, *Mobile Edge Computing*) são beneficiários em potencial dessa economia, e mesmo que seja desejável o suporte a implementação de funções genéricas nesses ambientes - devido a eventuais limitações de processamento - subconjuntos simplificados dos *frameworks* são suficientes e garantem a ampla utilização das plataformas nos mesmos.

4. Plataforma Click-on-OSv

Considerando o modelo de desenvolvimento apresentado na Seção 3, foi desenvolvido o protótipo de uma plataforma para execução de funções de rede virtualizadas, chamada Click-on-OSv. A arquitetura desta plataforma pode ser descrita através de três componentes executando sobre o sistema *unikernel* OSv: o acelerador de pacotes DPDK, o *framework* para desenvolvimento de funções de rede virtualizadas *Click Modular Router* (CMR) e um *Web Service* (WS) com interfaces REST para a realização da gerência. Cada um desses componentes atende um ou mais requisitos de criação de plataformas NFV, deste modo a utilização em conjunto dos mesmos permite uma implementação compatível a uma gama dos mesmos. Apesar da realização de modificações no CMR para suporte nativo ao NSH, o módulo Agente Estrutural não foi desenvolvido na versão protótipo da plataforma, dessa forma o Click-on-OSv conta com suporte a um único *framework* de desenvolvimento de funções de rede virtualizadas - o próprio CMR - e o processamento do NSH deve ser programado diretamente na função de rede. A Figura 3 apresenta os componentes que, integrados, compõem a plataforma desenvolvida.

A utilização de máquinas virtuais para a implantação de VNFs ou serviços com um alto custo computacional implica em problemas de desempenho, dado que existe a adição de uma camada de virtualização entre a VNF e o *hardware* em si. Com o objetivo de melhorar este desempenho, *hypervisors* implementam a técnica de paravirtualização, onde tarefas com alto custo de processamento podem ser realizadas com ajuda do sistema

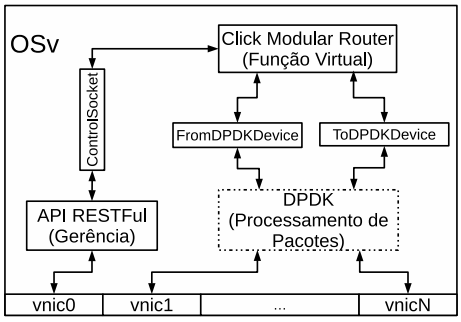

Figura 3. Arquitetura interna da plataforma desenvolvida

hospedeiro.

O *unikernel* OSv foi escolhido não apenas pelo seu suporte a diversos dispositivos paravirtualizados, mas também por ser um sistema simples, com um único espaço de endereçamento de memória (diminuindo a sobrecarga causada por trocas de contexto), possuir uma *Application Binary Interface* (ABI) compatível com a ABI Linux (importante pois *NFV Enablers* são majoritariamente desenvolvidos para sistemas *Unix-like*), e também uma interface REST facilmente extensível. Por ser um *unikernel*, o OSv disponibiliza apenas as funcionalidades necessárias para que uma aplicação específica possa ser executada, reduzindo assim seu consumo de recursos. Durante a inicialização da plataforma, o OSv utiliza parâmetros previamente definidos para inicializar e atribuir um endereço IP para a interface utilizada para gerência, repassa o controle das interfaces de rede que serão utilizadas pelo CMR para o DPDK e inicializa os componentes.

Embora seja opcional a utilização de um *framework* de aceleração de pacotes na arquitetura de uma VNF, a especificação [NFV-IFA 2016] apresenta entre tecnologias desejáveis a paravirtualização, uma vez que esta é capaz de realizar o compartilhamento de recursos (pois várias interfaces paravirtualizadas podem utilizar uma mesma interface física de forma isolada) e também o requisito de portabilidade (o modelo possui uma especificação bem definida já implementada por diferentes *hypervisors*). Desta forma, o DPDK foi escolhido como o *framework* para aceleração de pacotes por apresentar uma implementação do modelo de paravirtualização com suporte a *pooling* e pré-alocação de *buffers*. Na execução da plataforma, o DPDK desempenha funcionalidades tipicamente atribuídas a pilha de rede dos sistemas operacionais tradicionais.

Em relação ao *framework* para a execução das funções de rede virtualizadas, o *Click Modular Router* (CMR) apresentou-se como a opção mais adequada devido ao longo tempo de desenvolvimento, extensiva lista de componentes (elementos) que podem ser usados para a criação de funções virtualizadas de rede, disponibilidade de suporte ao DPDK, facilidade de gerência e recuperação de estatísticas e, por fim, ampla adoção em trabalhos relacionados [Martins et al. 2014] [Bu et al. 2018] - porém foi necessário realizar adições a biblioteca para fornecer elementos de alto nível para processamento e controle do NSH - chamado CMRvNSH. O CMRvNSH deve executar o processamento dos pacotes de acordo com a função virtual definida. Na plataforma, a comunicação entre o CMRvNSH e o DPDK é feita por elementos de entrada e saída, respectivamente, `FromDPDKDevice(DEVID)` e `ToDPDKDevice(DEVID)`, que constituem os únicos pontos de comunicação entre o ambos e, consequentemente, de comunicação com a rede

externa. A configuração das funções virtualizadas é definida através de um arquivo carregado na inicialização do CMRvNSH, que pode ser alterado pela API de gerência. Caso seja necessário modificar a função, deve-se substituir o arquivo e reinicializar os módulos da plataforma.

Por fim, para o gerenciamento das funções e da instância da plataforma a API REST nativa do OSv é utilizada, por esta já disponibilizar um conjunto de métodos úteis (*e.g.*, reiniciar a instância, upload de arquivos e finalizar aplicações). Também foram criadas extensões para esta API, tomando como base os requisitos de gerência investigados em [Bondan et al. 2014], sendo estas extensões relacionadas ao monitoramento de estatísticas da VNF através da utilização de um `ControlSocket` do CMR, e a reconfiguração das funções, que pode ser feita através do upload de uma nova função seguida da reinicialização da dos módulos da instância. A escolha do protocolo de gerência foi determinado através da análise da especificação ETSI, que define um modelo com funções necessárias para a configuração e gerência de VNFs. Como a interface de gerência utilizada nesta plataforma deve disponibilizar apenas o controle do ciclo de vida da sua própria instância, a interface desenvolvida é capaz de disponibilizar métodos para serem utilizados pelo VNFM para controlar a instância, atuando como um EMS.

4.1. Validação e Avaliação

A plataforma desenvolvida foi validada através de cenário de testes análogo ao definido na RFC 2544 [Bradner and McQuaid 1999], sendo composto por duas máquinas virtuais (VM) funcionando como transmissor e receptor, e uma máquina virtual executando as plataformas a serem testadas. O sistema operacional onde as VMs são executadas consiste do sistema operacional Debian com *hypervisors* Xen 4.4.2 e KVM 1.2. O servidor de execução do *hypervisor* possui processador Intel Xeon E5620@2.40Ghz com 4 núcleos e 12GB de memória DDR3-ECC@1066MHz. Para o receptor e o transmissor foram dedicados 1GB de memória RAM e um núcleo físico, garantido através de CPU *Pinning*.

As plataformas selecionadas para testes, totalizando quatro cenários, são: ClickOS, Linux, Xen Click-on-OSv e KVM Click-on-OSv. Para o ClickOS e Click-on-OSv 192MB de memória RAM e um núcleo físico. Devido seus requisitos mínimos de recursos, o Linux conta com 1GB de memória e um núcleo físico. Para a interconexão das VMs, o comutador virtual OpenVSwitch foi utilizado tanto no Xen quanto no KVM, conectados através de duas pontes, sendo uma para a comunicação do transmissor com a plataforma e a outra para comunicação da plataforma com o receptor. Todas as VMs foram configuradas para utilizarem a paravirtualização padrão do seu respectivo *hypervisor*(netfront para Xen e VirtIO para KVM - a exceção do Xen Click-on-OSv onde VirtIO foi utilizado por questões de compatibilidade). Devido a falta de suporte, o ClickOS não faz uso das funcionalidades *Generic Segmentation Offload* (GSO), *TCP Segmentation Offload* (TSO) e *Generic Receive Offload* (GRO) em suas interfaces.

A função de rede utilizada para avaliação é responsável por encaminhar pacotes entre duas interfaces de rede, sem realizar nenhum processamento extra. Tal função foi escolhida por possibilitar que o desempenho máximo teórico em ambas as plataformas fosse observado. Além disso, como definição de um cenário ótimo, a máquina virtual executando Linux apenas cria uma ponte interna entre as duas interfaces utilizando *linux-bridges*.

As ferramentas definidas para a realização das medições foram o *netperf* para a medida de vazão, com um fluxo TCP contínuo com pacotes variando de 64B a 1500B, sendo realizado no final de cada bateria de testes a média da vazão dos pacotes, e o D-ITG para a medida de atraso, com um fluxo UDP simulando uma conexão VoIP, um fluxo TCP simulando uma conexão Telnet, e um fluxo UDP com uma variação aleatória no tamanho dos datagramas entre 64 e 1500 Bytes. Estes três fluxos foram escolhidos por apresentarem comportamento semelhante ao de aplicações normalmente utilizadas em uma rede de produção. O atraso foi calculado através do *Round Trip Time* devido a sincronização imprecisa entre o transmissor e receptor. Todos os testes foram realizados 30 vezes, de modo a atingir um intervalo de confiança de 95%.

As métricas utilizadas na avaliação e na comparação do Click-on-OSv com as demais plataformas foram definidas também de acordo com a RFC 2544 [Bradner and McQuaid 1999]. É importante ressaltar que alguns parâmetros de testes definidos pela RFC não se enquadram em cenários de NFV como, por exemplo, tamanho de *frames* para FDDI e *Token Ring*. Assim, as métricas mais utilizadas para a avaliação de dispositivos de rede costumam ser vazão e atraso.

A Figura 4 apresenta o gráfico de vazão das plataformas testadas. Tomando o Linux como o valor máximo possível neste cenário, o Click-on-OSv demonstra bons resultados quando utilizado em conjunto com o KVM, e consistentemente possui um desempenho superior em relação ao ClickOS. A média da vazão para pacotes de 128B do ClickOS é superior ao da plataforma desenvolvida, porém como a diferença está dentro da variação verificada, constata-se que as duas possuem um desempenho semelhante.

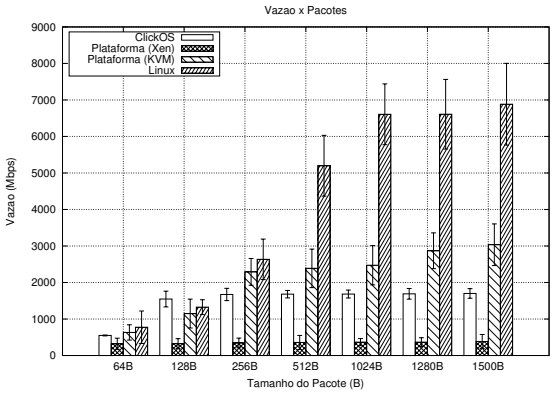

Figura 4. Avaliação da vazão das plataformas

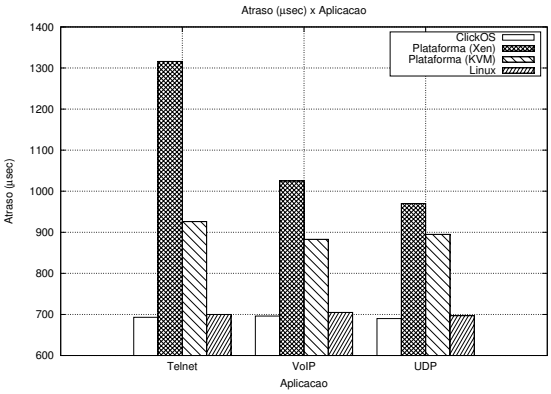

Figura 5. Avaliação do atraso nas plataformas

A Figura 5 apresenta o atraso das plataformas testadas. É possível observar que o ClickOS possui um atraso semelhante ao do Linux para os três casos, enquanto o Click-on-OSv apresenta medidas atraso superiores. Mesmo assim, em todos os testes, exceto no Click-on-OSv utilizado em conjunto com o Xen, apresentam valores inferiores a 1 milissegundo.

A vazão do Click-on-OSv a é superior ao do ClickOS quando executada no KVM devido a otimização tanto do OSv quanto do DPDK para seu funcionamento com as interfaces paravirtualizadas VirtIO. Em relação ao atraso, esperava-se que o ClickOS obtivesse um resultado menor que Click-on-OSv devido ao uso do Mini-OS que, embora não seja

otimizado em termos de vazão, apresenta um atraso reduzido devido ao seu compartilhamento de uma área de memória com o *hypervisor*.

No entanto, o desempenho inferior verificado nas métricas para o Click-on-OSv executando no *hypervisor* Xen foi considerado um resultado inesperado. Esse resultado é atribuído a implementação de paravirtualização VirtIO, esta foi realizada para efeitos de compatibilidade, e não focada na otimização de desempenho, como verificado no KVM. Assim, uma alternativa para contornar este baixo desempenho passa pela implementação *drivers* no DPDK para a plataforma *netfront*, utilizado nativamente pelo Xen, ou utilizar diretamente a pilha de rede do OSv, uma vez que esta possui suporte ao *netfront*.

Apesar da deficiência do Click-on-OSv em relação ao atraso, existem vantagens quanto aos requisitos funcionais de NFV. A exemplo disso é a sua capacidade de executar em diferentes *hypervisors* sem modificações, fazendo-se adequada para cenários onde existem infraestruturas de diferentes fornecedores e requisitos com relação ao atraso não sejam absolutamente estritos, uma vez que a diferença de desempenho neste requisito é inferior a meio milissegundo. Ademais, a interface de gerência do Click-on-OSv também é outro ponto positivo, considerando que a gerência de diversas instâncias pode ser centralizada em um único ponto através da interface REST, o que não é possível em seu concorrente mais próximo, o ClickOS.

5. Conclusão

A área de virtualização de funções de rede tem atraído grande interesse de pesquisadores do mundo inteiro. Por ainda se encontrar em um estágio inicial, alguns conceitos e métodos não estão definidos claramente. As plataformas para executar VNFs são um destes, já que, embora existam modelos e especificações apresentando requisitos de alto nível, ainda não existe um modelo de desenvolvimento sólido e nem mesmo uma implementação capaz de atender efetivamente os requisitos já especificados.

Este artigo apresenta uma proposta de modelo de desenvolvimento de plataformas para a execução de funções de rede virtualizadas. Através do detalhamento dos componentes necessários para implementação eficiente, respeitando os requisitos operacionais das VNFs, uma visão arquitetural destinada a esse tipo de plataforma foi concretizada. Juntamente, um protótipo, intitulado Click-on-OSv, foi desenvolvido compreendendo a maior parte dos módulos previstos e seu desempenho comparado a uma segunda plataforma com características semelhantes.

Em trabalhos futuros objetiva-se determinar a adequabilidade de diferentes tecnologias já existentes atuando como NFV *Enablers*, apontando cenários de aplicação onde uma ou outra apresenta comportamento de destaque e dessa forma facilitar a escolha das mesmas para implementações de plataformas direcionadas a um determinado ambiente ou função. Além disso, pretende-se realizar aprimoramentos na plataforma Click-on-OSv como a implementação do agente estrutural e a disponibilização de novos *frameworks* de desenvolvimento de funções de rede virtuais, concretizando a mesma como uma opção completa e genérica para execução de VNFs. Finalmente, projeta-se a definição de documentos delineando diretivas arquiteturais e de implementação ricamente detalhadas, além de desafios e oportunidades relacionados a plataformas para execução de funções de rede virtualizadas.

Referências

Bernardos, C. J., Rahman, A., Zúñiga, J.-C., Contreras, L. M., Aranda, P. A., and Lynch, P. (2018). Network Virtualization Research Challenges. Internet-Draft draft-irtf-nfvrg-gaps-network-virtualization-09, Internet Engineering Task Force. Work in Progress.

Blumenthal, U. and Wijnen, B. (2002). User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414.

Bondan, L., d. Santos, C. R. P., and Granville, L. Z. (2014). Management requirements for clickos-based network function virtualization. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 447–450.

Bradner, S. and McQuaid, J. (1999). Benchmarking methodology for network interconnect devices. RFC 2544, RFC Editor.

Bu, C., Wang, X., Huang, M., and Li, K. (2018). Sdnfv-based dynamic network function deployment: model and mechanism. *IEEE Communications Letters*, 22(1):93–96.

ETSI (2012). Network functions virtualisation – introductory white paper. Acesso em 21 Abr. 2018.

Halpern, J. M. and Pignataro, C. (2015). Service Function Chaining (SFC) Architecture. RFC 7665.

Handley, M. (2006). Why the internet only just works. *BT Technology Journal*, 24(3):119–129.

Intel, D. (2014). Data plane development kit. *URL http://dpdk. org*.

ISG, N. E. (2013). Etsi gs nfv 004 v1. 1.1-2013 network functions virtualisation (nfv): Virtualisation requirements.

Levi, D. B. and Schönwälder, J. (2001). Definitions of Managed Objects for the Delegation of Management Scripts. RFC 3165.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). Clickos and the art of network function virtualization. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 459–473, Berkeley, CA, USA. USENIX Association.

NFV-IFA (2016). Network functions virtualisation; acceleration technologies; vswitch benchmarking and acceleration specification. *ETSI, April*.

Quinn, P., Elzur, U., and Pignataro, C. (2018). Network Service Header (NSH). RFC 8300.

Rizzo, L. (2012). netmap: A novel framework for fast packet i/o. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 101–112, Boston, MA. USENIX Association.

Taylor, D. and Turner, J. (2004). Towards a diversified internet. *White paper, November*.

Zhang, W., Liu, G., Zhang, W., Shah, N., Lopreiato, P., Todeschi, G., Ramakrishnan, K., and Wood, T. (2016). Opennetvm: A platform for high performance network service chains. In *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMIddlebox '16, pages 26–31, New York, NY, USA. ACM.