NFV-Prime: Uma Ferramenta para Prototipação e Visualização de Funções Virtualizadas de Rede

Felipe Ribeiro Quiles **Vinicius Fulber-Garcia** **Elias P. Duarte Jr.**

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19018 Curitiba 81531-990 PR

{frquiles, vinicius, elias}@inf.ufpr.br

Resumo. *O paradigma NFV (Network Functions Virtualization) vem se popularizando nas redes modernas. Este trabalho apresenta a ferramenta NFV-Prime que permite a prototipação de VNFs (Virtual Network Functions) de forma simples e intuitiva. A ferramenta integra quatro componentes em um pipeline: Editor, Gerenciador de Interfaces Virtuais de Rede, Gerador de Tráfego e Visualização de Resultados. A NFV-Prime permite a implementação de VNFs em Python 3, além de oferecer uma ampla variedade de parametrizações e configurações das interfaces de rede e dos tráfegos gerados durante a execução da VNF. Para demonstrar a ferramenta, é apresentada a criação de um balanceador de tráfego stateful.*

1. Introdução

Redes 5G, redes de *datacenter* e a própria Internet fazem uso de Funções de Rede (*Network Functions* - NF) para a execução das mais diversas tarefas. *Firewalls*, dispositivos NAT (*Network Address Translation*) e sistemas de detecção de intrusão são exemplos de NFs. O paradigma de Virtualização de Funções de Rede (*Network Functions Virtualization* - NFV) viabiliza a implementação de NFs em software por meio do uso de tecnologias de virtualização. A *European Telecommunications Standards Institute* (ETSI) propôs a arquitetura de referência NFV-MANO (*NFV Management and Orchestration*) [ETSI 2024a] para orientar a implementação e integração de soluções NFV. Além disso, o paradigma NFV tem sido explorado no contexto de *in-network computing* [Venâncio et al. 2022], permitindo a execução de tarefas genéricas dentro da rede (exemplos incluem [Turchetti and Duarte 2015, Venâncio et al. 2019]).

A arquitetura de referência MANO define que o ambiente NFV [ETSI 2024b] é composto por funções de rede executando em uma Infraestrutura Virtualizada (*Virtualized Infrastructure* - VI) e sendo gerenciadas por outros componentes da arquitetura. Funções individuais também podem ser orquestradas para a construção de serviços virtuais complexos [Fulber-Garcia et al. 2020a, Fulber-Garcia et al. 2024], os quais podem estar distribuídos entre diversos sistemas autônomos [Huff et al. 2020, Fulber-Garcia et al. 2021]. Atualmente, diversos sistemas implementam o modelo NFV-MANO, com destaque para o OpenStack Tacker [OpenStack 2016], OSM [ETSI 2020], OpenBaton [Fraunhofer-Fokus and Tu-Berlin 2017], Vines [Flauzino et al. 2021], entre outros. Um componente essencial desses sistemas são as plataformas de execução de VNFs, que fornecem os recursos necessários para a execução de funções de rede. Exemplos de plataformas de execução de VNFs incluem ClickOS [Martins et al. 2014],

Click-On OSv [Marcuzzo et al. 2017], COVEN [Fulber-Garcia et al. 2019] e Mini-NFV [Castillo-Lema et al. 2019].

Sem exceção, as plataformas citadas anteriormente possuem ambientes de difícil instalação, sendo particularmente desafiadoras para novos usuários, especialmente aqueles iniciando na programação de funções virtualizadas de rede. Muitas vezes, exigem que múltiplos sistemas distintos sejam instalados e configurados para poderem interagir entre si. Para solucionar esse problema, o presente trabalho propõe a ferramenta NFV-Prime, construída visando introduzir, de forma prática e facilitada, o paradigma NFV para novos desenvolvedores e usuários. A NFV-Prime simplifica o processo de criação, instanciação e visualização dos resultados de execução das VNFs em uma rede de testes.

A NFV-Prime é de fácil instalação e apresenta uma interface simples que permite ao usuário dominar rapidamente todo o ciclo de vida de uma VNF. A ferramenta consiste em quatro grandes componentes funcionais: Editor, Gerenciador de Interfaces Virtuais de Rede, Gerador de Tráfego e Visualização de Resultados. Os componentes têm o objetivo de serem didáticos, criando um *pipeline* de configuração e execução intuitivo para os usuários, visando reduzir a probabilidade de erros nos processos de configuração e execução de VNFs. Neste artigo, a arquitetura da ferramenta é descrita, incluindo os módulos construídos para tarefas como configuração de tráfego, gerenciamento de interfaces virtuais de rede e monitoramento do ambiente. Os módulos foram projetados para ter baixo acoplamento entre si, o que permite sua evolução natural conforme o desenvolvimento e surgimento de novas tecnologias e ferramentas.

Para demonstrar a ferramenta, este artigo descreve a construção e execução de uma VNF de balanceamento de carga em modo *stateful*. Diversos outros estudos de caso são apresentados em [Quiles 2024] e não foram incluídos no artigo devido a limitações de espaço. Pelo mesmo motivo, não foram incluídas as avaliações qualitativas realizadas para demonstrar a capacidade de redução de erros nos processos de configuração e execução de VNFs. Nesse sentido, a ferramenta NFV-Prime foi avaliada em três cenários distintos, através da utilização do modelo *Human Error Assessment and Reduction Technique* (HEART) [Alexander 2017]. Foi possível aferir uma redução de 66,6% no número de casos possíveis de erros, além de uma significativa redução no impacto dos erros remanescentes no gerenciamento do ciclo de vida de uma VNF, quando comparado ao uso de um sistema NFV-MANO genérico.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados; a Seção 3 descreve a arquitetura e a Seção 4 apresenta a implementação da NFV-Prime; a Seção 5 apresenta um estudo de caso com um balanceador de carga *stateful*; por fim, a conclusão é apresentada na Seção 6.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados à ferramenta NFV-Prime. Destacamos que nenhum dos trabalhos relacionados tem o propósito e escopo exatos da ferramenta NFV-Prime, que visa ser uma ferramenta simples, guiada e assistida para o desenvolvimento e teste de funções de rede virtualizadas. A ferramenta Click-on-OSv [Marcuzzo et al. 2017] permite a instanciação e execução de *middleboxes* baseadas no roteador virtual *Click Modular Router*. Para utilizar a plataforma, o usuário desenvolve a função de rede no Click-on-OSv, que a instancia e executa, coletando informações sobre o funcionamento,

as quais são mostradas por meio de gráficos e estatísticas. A plataforma é disponibilizada via uma imagem de máquina virtual baseada no sistema operacional minimalista OSv. O *framework* de programação disponibilizado é o do próprio *Click Modular Router*, de uso bastante desafiador por ser exclusivo para este sistema.

[Tavares et al. 2018] propõe a plataforma *NFV Infrastructure Emulation Platform* (NIEP) para o desenvolvimento e teste de VNFs e infraestruturas baseadas no paradigma NFV. O NIEP foi construído utilizando o emulador de rede Mininet em conjunto com a plataforma de desenvolvimento de VNFs Click-on-OSv. O NIEP visa prover ao usuário um ambiente de emulação NFV completo [Fulber-Garcia et al. 2020b], permitindo a execução de testes utilizando as VNFs desenvolvidas em um cenário controlado antes de colocá-las em ambientes de produção.

[Peuster et al. 2016] apresentam uma plataforma de prototipagem NFV, chamada *Multi Datacenter servIce ChaIN Emulator* (MeDICINE), que visa executar NFs em um ambiente Multi-PoP (*Multi-Points of Presence*). As NFs são disponibilizadas por meio de contêineres. Por meio de sistemas de gerenciamento e orquestração conectados à plataforma por meio de interfaces padronizadas, é possível gerenciar as NFs. O MeDICINE tem como base a ferramenta Containernet, que estende a estrutura de emulação do Mininet e permite a utilização de contêineres padronizados como instâncias de computação dentro da rede emulada.

O Containernet 2.0 [Peuster et al. 2018] é uma plataforma de prototipagem NFV que visa auxiliar na criação e execução, localmente, de SFCs. A plataforma se destaca por suportar SFCs híbridas, sendo possível que a cadeia seja composta por VNFs baseadas em contêineres e VNFs baseadas em máquinas virtuais. A instanciação de máquinas virtuais ocorre através da utilização de um *script* em Python.

O *framework* Mini-NFV [Castillo-Lema et al. 2019] para Orquestração NFV é um Gerenciador de VNF de uso geral para implementar e operar funções de rede virtuais e serviços de rede no Mininet utilizando especificações TOSCA. O objetivo principal do Mini-NFV é remover do programador a tarefa de configurar todo o ambiente de encadeamento de serviços, permitindo que ele possa dedicar seu tempo inteiramente ao desenvolvimento das VNFs. Além disso, o uso do Mini-NFV facilita a transição de emulação para o mundo real, reforçando a replicabilidade e reprodutibilidade do paradigma NFV.

3. NFV-Prime: Arquitetura

A arquitetura da ferramenta NFV-Prime foi proposta com o objetivo de ser genérica, abrangente e expansível. A Figura 1 ilustra a arquitetura, constituída por nove módulos, descritos nos próximos parágrafos: (i) Interface de Acesso (IA), (ii) Interface Gráfica de Usuário (GUI), (iii) Gerenciador de Acesso (GA), (iv) Gerenciador da Plataforma (GP), (v) Gerenciador do Banco de Dados (GBD), (vi) Monitor (M), (vii) Gerenciador da Rede (GR), (viii) Gerenciador de Tráfego (GT) e (ix) Aplicação (App). Cada módulo é responsável por executar operações específicas de tratamento, encaminhamento e/ou aplicação de requisições, sejam essas referentes à instanciação da aplicação, configuração de tráfego, gerenciamento de interfaces virtuais de rede e monitoramento da plataforma - representados por setas contínuas - ou pela relação entre as funcionalidades dos componentes, ou seja, a interação que ocorre entre um ou mais módulos da plataforma - representadas por linhas tracejadas.

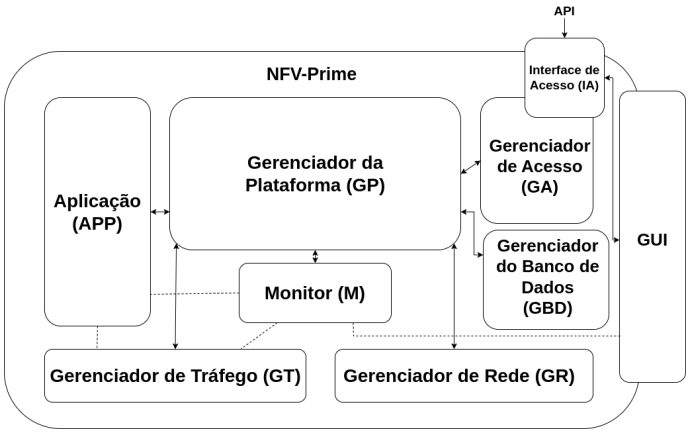

Figura 1. Arquitetura da ferramenta NFV-Prime.

Interface de Acesso: é o módulo responsável pelo recebimento de requisições que podem vir da API da NFV-Prime ou da Interface Gráfica de Usuário (GUI), que por sua vez consome a API. Ao receber uma requisição, a Interface de Acesso realiza uma breve análise sintática, filtrando e descartando requisições que estejam incompletas ou contenham erros em sua construção. Se a requisição estiver adequada à API, é encaminhada para o Gerenciador de Acesso.

Interface Gráfica de Usuário (GUI): é o módulo responsável por apresentar a NFV-Prime ao usuário, fornecendo uma interface gráfica que se comunica com outros módulos da plataforma. A requisição construída pela GUI é encaminhada ao módulo Interface de Acesso. O GUI apresenta ao usuário as quatro principais funcionalidades da NFV-Prime: edição, gerenciamento de interfaces virtuais de rede, parametrização e geração de tráfego de rede, e monitoramento de interfaces e VNFs.

Gerenciador de Acesso: é o módulo que recebe as requisições da Interface de Acesso. O Gerenciador de Acesso é responsável pela reorganização das requisições, possibilitando que elas sejam interpretadas pelo módulo Gerenciador da Plataforma. Ele realiza a separação entre função e parâmetros. Por fim, o Gerenciador de Acesso reencaminha as requisições de forma organizada e simplificada para o Gerenciador da Plataforma.

Gerenciador da Plataforma: é o módulo responsável, principalmente, pelo ciclo de vida dos demais módulos da plataforma NFV-Prime: Gerenciador do Banco de Dados, Monitor, Gerenciador de Rede, Gerenciador de Tráfego e Aplicação. O módulo recebe e interpreta as requisições do Gerenciador de Acesso, identifica o módulo responsável pela funcionalidade solicitada e encaminha um sinal de início ou parada. As ações indicadas podem envolver a instanciação ou remoção de recursos e VNFs.

Gerenciador do Banco de Dados: é o módulo responsável pelo armazenamento, atualização, remoção e controle dos dados da plataforma. Esse módulo mantém uma relação estreita com o Gerenciador da Plataforma, do qual recebe dados para serem armazenados ou atualizados no banco para posterior acesso.

Monitor: é responsável por monitorar os módulos de Aplicação, Gerenciador de Rede e Gerenciador de Tráfego, utilizando VNFs. O Monitor coleta dados acerca dos pro-

cessos em execução nesses módulos e repassa essas informações ao módulo Gerenciador da Plataforma, que, por sua vez, aciona o módulo Gerenciador do Banco de Dados para atualizar e armazenar os dados coletados.

Gerenciador de Rede: a principal funcionalidade do Gerenciador de Rede é, quando requisitado pelo Gerenciador da Plataforma, instanciar, remover e manter interfaces virtuais de rede. A partir da disponibilização de uma interface na ferramenta, ela pode começar a receber tráfego de rede. Existe uma interação entre o Gerenciador de Rede, o Gerenciador de Tráfego e a Aplicação.

Gerenciador de Tráfego: é o módulo responsável pela configuração e geração de tráfego sintético. Sua principal funcionalidade é a inicialização e interrupção do fluxo de rede. O Gerenciador de Tráfego recebe do Gerenciador da Plataforma a parametrização para a geração do fluxo, com parâmetros definidos pelo usuário por meio da GUI ou diretamente via API da NFV-Prime.

Aplicação: é o módulo responsável pelo gerenciamento das VNFs desenvolvidas pelo usuário. Sua principal funcionalidade é instanciar e remover VNFs na ferramenta conforme as requisições do Gerenciador da Plataforma. A VNF desenvolvida é enviada ao Gerenciador da Plataforma por meio da API da NFV-Prime, acessível diretamente ou via GUI. Após a interpretação da requisição e a identificação da necessidade de intervenção do módulo de Aplicação, este é acionado. Assim, a ação solicitada, seja a instanciação ou remoção da VNF na rede, é executada imediatamente.

De maneira geral, quando uma requisição originada pela GUI ou API é recebida pela Interface de Acesso, ocorre uma filtragem para descartar requisições que não estejam no formato correto. Caso estejam corretas, a Interface de Acesso encaminha a requisição ao Gerenciador de Acesso, que verifica se a operação solicitada está entre as funcionalidades disponíveis na plataforma e se os parâmetros são adequados. Se a requisição for válida e autorizada, é encaminhada ao Gerenciador da Plataforma, que a interpreta e, se necessário, consulta o Gerenciador do Banco de Dados para obter informações complementares. Em seguida, o Gerenciador da Plataforma determina qual dos módulos finais — Monitor, Gerenciador de Rede, Gerenciador de Tráfego ou Aplicação — será responsável por executar a funcionalidade solicitada e a repassa ao módulo correspondente. O módulo final, ao receber a requisição, a executa imediatamente e, se necessário, retorna os dados ao Gerenciador da Plataforma, que pode acionar novamente o Gerenciador do Banco de Dados para atualização e armazenamento dessas informações. Os resultados das requisições seguem o caminho inverso, retornando até a GUI, que os utiliza conforme necessário.

3.1. Implementação

A ferramenta NFV-Prime é uma aplicação Web, estratégia escolhida para facilitar o acesso e ampliar a divulgação tanto da ferramenta quanto do próprio paradigma NFV[1]. O desenvolvimento priorizou uma interface amigável, com potencial para acelerar a criação e instanciação de VNFs. Um dos principais objetivos da ferramenta é incentivar e simplificar a exploração das possibilidades de implementação de NFs proporcionadas pelo paradigma.

[1]O código-fonte da plataforma está disponível em `https://github.com/fequiles/NFV-Prime.git`

A ferramenta NFV-Prime foi estruturada em quatro componentes principais: Editor, Gerenciador de Interfaces Virtuais de Rede, Gerador de Tráfego e Visualização. Esses componentes, na ordem apresentada, formam um *pipeline* para a elaboração e execução de NFs. Além disso, a NFV-Prime permite a importação e exportação de cenários em arquivos JSON, facilitando a reutilização de funções de forma rápida e prática. O componente Editor possibilita ao usuário desenvolver a VNF diretamente na ferramenta ou importar seu código, que deve ser obrigatoriamente escrito em Python 3. A aplicação Web foi implementada utilizando a biblioteca React[2].

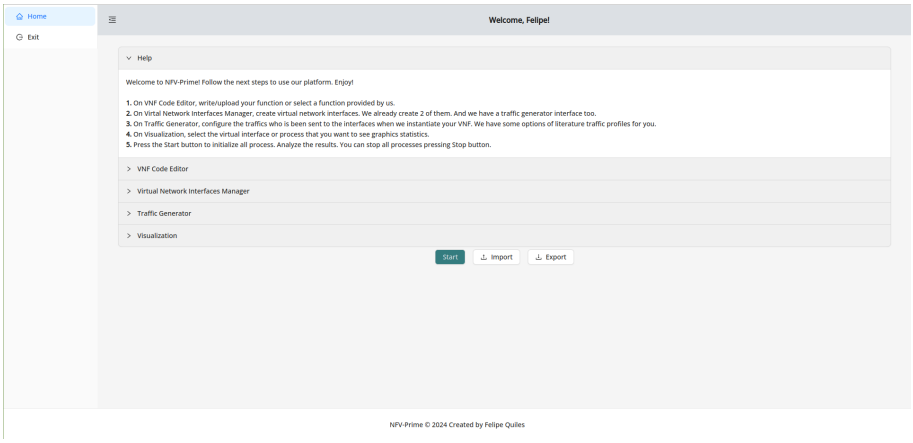

Figura 2. Interface da ferramenta NFV-Prime.

O Gerenciador de Interfaces Virtuais de Rede permite ao usuário controlar a quantidade de interfaces de rede presentes na NFV-Prime. Por padrão, a NFV-Prime inicia com três interfaces virtuais de rede instanciadas: *dummy_1* e *dummy_2*, e a terceira, *dummy_traffic_generator*, responsável pela geração de tráfego na rede. O usuário pode adicionar ou remover novas *dummies*. Para a criação de interfaces, foi utilizado o utilitário netns[3], que cria um novo ambiente de rede isolado dentro da mesma máquina. O ambiente principal (*host*) é conectado ao ambiente gerado pelo netns (*net namespace*), representando os *endpoints* da rede. Na NFV-Prime, foram criados dois ambientes: um para o gerador de tráfego, responsável pela *dummy* geradora de tráfego, e outro para as demais *dummies*.

O Gerador de Tráfego possibilita a configuração e parametrização de tráfegos de rede sintéticos, que serão executados na rede em conjunto com a VNF desenvolvida pelo usuário. A geração do tráfego na NFV-Prime pode ocorrer de duas formas. A primeira é o modo automático, que utiliza a ferramenta *Nping* (versão 0.7.8) e permite ao usuário parametrizar o tráfego gerado. Para auxiliar nesse processo, a ferramenta disponibiliza dois perfis de tráfego pré-configurados: elefante [Hamdan et al. 2020] e guepardo [Maji et al. 2017]. A segunda opção é o modo manual, no qual o usuário pode desenvolver e instanciar seu próprio gerador de tráfego, utilizando a linguagem Python 3. Assim como no componente Editor, também é possível importar arquivos para a configuração do gerador de tráfego.

Ao finalizar a configuração de um tráfego, o usuário deve adicioná-lo à lista de

[2]`https://react.dev`
[3]`https://github.com/Lekensteyn/netns`

tráfegos para ser gerado na rede quando a ferramenta NFV-Prime executar o *pipeline*. Vale ressaltar que, enquanto a ferramenta NFV-Prime não estiver executando a VNF na rede, é possível editar ou remover os tráfegos presentes na lista de disponíveis.

O componente de Visualização é responsável por exibir, em tempo real, estatísticas da interface virtual de rede selecionada. São apresentados ao usuário, em um gráfico, a taxa de transmissão (TX) e de recebimento (RX) da interface de rede. Os gráficos permitem visualizar os tráfegos de entrada e saída, demonstrando o funcionamento da VNF em execução. Dessa forma, é possível analisar, em tempo real, os impactos que diferentes mecanismos de rede podem gerar no tráfego quando instanciados em diversos cenários e parametrizações. Além disso, também é possível visualizar a demanda da VNF em termos de uso de memória (MEM) e processador (CPU).

4. NFV-Prime: Avaliação

A fim de avaliar a plataforma NFV-Prime, foi realizada uma comparação entre a interface proposta e a implementação de uma arquitetura NFV-MANO genérica [Fulber-Garcia et al. 2020b]. O método utilizado para essa comparação foi o modelo *Human Error Assessment and Reduction Technique* (HEART) [Alexander 2017], que busca determinar o nível de incerteza associado a uma tarefa genérica. Para demonstrar a capacidade de redução de erros nos processos de configuração e execução de VNFs, a plataforma NFV-Prime foi avaliada, em três cenários distintos, utilizando esse modelo. Os resultados indicaram uma redução de 66,6% no número de casos possíveis de erros, além de uma significativa diminuição no impacto dos erros remanescentes no gerenciamento do ciclo de vida de uma VNF, em comparação ao sistema NFV-MANO genérico. Estes resultados estão disponíveis na referência [Quiles 2024] e não foram incluídos neste artigo devido às limitações de espaço.

Para demonstrar a ferramenta, é apresentado um estudo de caso no contexto de engenharia de tráfego, utilizando o mecanismo de *Stateful Load Balancer* (SLB)[4]. O SLB tem como objetivo balancear os fluxos de rede recebidos de forma uniforme entre os diferentes *endpoints* disponíveis. A VNF, ao receber um pacote, realiza seu desempacotamento e verifica se o IP de destino está nos padrões das interfaces do NFV-Prime. Em caso afirmativo, verifica-se, utilizando a chave do fluxo, o IP de destino e as portas de origem e destino, se o pacote já está sendo direcionado para uma das interfaces; se estiver, o pacote do fluxo é encaminhado imediatamente para a interface correspondente. No entanto, se o pacote analisado for o primeiro do fluxo, um contador de fluxos é utilizado para determinar qual das interfaces disponíveis passará a receber o fluxo. Para realizar este estudo de caso, foram utilizadas quatro interfaces virtuais de rede, *dummies* de 1 a 4, sendo o tráfego enviado para a *dummy_4*. Também houve a parametrização dos tráfegos de rede, totalizando três configurações diferentes, sendo duas, nomeadas de guepardo e "middle", utilizando o *Nping*, enquanto a outra é um gerador de tráfego manual.

Os parâmetros de tráfego utilizados para o guepardo foram: taxa de transmissão de 100 (pacotes por segundo - p/s), pacotes com 64 *bytes*, transmissão de 500 pacotes, porta de destino 8005, sem atraso nos pacotes e com gatilho de inicialização de 3000 milissegundos (ms). O tráfego "middle" tem a seguinte configuração: taxa de transmissão

[4]Os experimentos estão disponíveis em `https://github.com/fequiles/NFV-Prime/tree/master/NFVPrimeExemplos`

de 25 (p/s), pacotes com 256 *bytes*, transmissão de 250 pacotes, porta de destino 8004, sem atraso e com gatilho de inicialização de 1500 ms. Já o tráfego manual gera 3 tipos de tráfego, todos realizando o envio de 1000 pacotes, sendo que os 3 tráfegos têm as seguintes configurações, respectivamente: taxa de transmissão igual a 100, 25 e 60 (p/s), pacotes com 16, 1024 e 256 *bytes*, e porta de destino 8001, 8002 e 8003.

Ao final da execução da VNF, a distribuição dos pacotes ficou da seguinte forma: a *dummy_1* 33,3%, *dummy_2* 40% e *dummy_3* 26,7% dos pacotes. A Figura 3 apresenta a taxa média do RX das *dummies* a cada segundo da execução da VNF. Percebe-se a variação do RX médio quando um fluxo é encerrado ou seu recebimento é iniciado pela interface de rede, sendo possível notar esse comportamento tanto na *dummy_1* (no segundo 1.5) quanto na *dummy_2*. Na *dummy_2*, é possível notar que, a partir do terceiro segundo, a interface começou a receber mais de um fluxo, com um aumento no valor da RX de 25.533 *bytes* para 29.956 *bytes* (em média no 3º e 4º segundo, respectivamente).

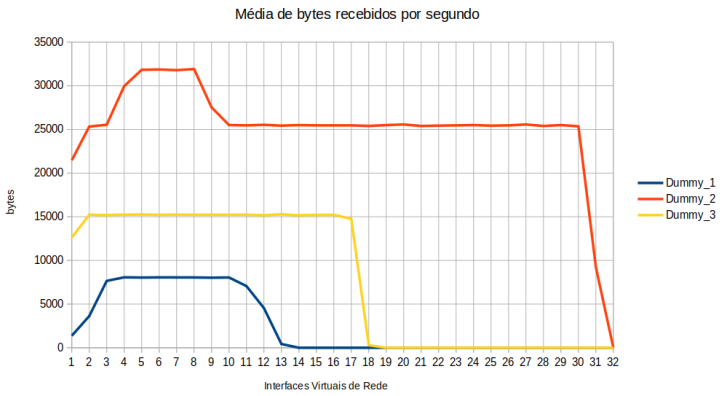

Figura 3. Gráfico da média de *bytes* recebidos por segundo pelas interfaces.

Em síntese, o estudo de caso demonstrou a atuação do mecanismo de balanceamento de carga, sendo possível visualizar nos gráficos da ferramenta a execução e a funcionalidade da VNF.

5. Conclusão

Este trabalho apresentou a ferramenta NFV-Prime para prototipação e visualização de VNFs, de forma simples e intuitiva, com o potencial de facilitar a introdução ao tema para novos usuários. A NFV-Prime consiste em quatro componentes: Editor, Gerenciador de Interfaces Virtuais de Rede, Gerador de Tráfego, além do componente de Visualização, que permite acompanhar a execução da função de rede em tempo real. Vale reforçar que a plataforma NFV-Prime possibilita implementar VNFs arbitrárias em Python3, além de permitir uma grande variedade de parametrizações e configurações de interfaces e tráfegos de rede que serão gerados durante a execução da VNF. Para a apresentação da plataforma, foram desenvolvidos estudos de caso no contexto de engenharia de tráfego.

Trabalhos futuros incluem expandir a plataforma NFV-Prime, permitindo múltiplos usuários simultâneos. Nesse caso, será necessário também ampliar o serviço de autenticação. Além disso, está prevista a expansão das métricas do módulo Visualização. Também deverá ser criado um componente de *debugging* da plataforma, que permitirá ao usuário verificar e analisar a execução da VNF por meio desse componente.

Agradecimentos

Os autores agradecem ao Programa de Pós-Graduação em Informática (PPGInf) da Universidade Federal do Paraná (UFPR) e ao PROEX CAPES por financiarem parcialmente este artigo, bem como ao CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, projeto número 308959/2020-5.

Referências

Alexander, T. M. (2017). Human error assessment and reduction technique (heart) and human factor analysis and classification system (hfacs). In *Collaboration on Quality in the Space and Defense Industries Forum*.

Castillo-Lema, J., Neto, A. V., de Oliveira, F., and Kofuji, S. T. (2019). Mininet-nfv: Evolving mininet with oasis tosca nvf profiles towards reproducible nfv prototyping. In *2019 IEEE Conference on Network Softwarization*, pages 506–512. IEEE.

ETSI (2020). Osm. `https://osm.etsi.org/`. Acessado em 08/04/2025.

ETSI (2024a). Network functions virtualisation (nfv) release 4; management and orchestration; architectural framework specification. *`https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV006v4.5.1-GS-MANOArchFwk.pdf`*.

ETSI (2024b). Network functions virtualisation (nfv); terminology for main concepts in nfv. *`https://www.etsi.org/deliver/etsi_gr/NFV/001_099/003/01.09.01_60/gr_NFV003v010901p.pdf`*.

Flauzino, J., Fulber-Garcia, V., Huff, A., Venâncio, G., and Duarte Jr, E. P. (2021). Gerência e orquestração de funções e serviços de rede virtualizados em nuvem cloudstack. In *Workshop de Gerência e Operação de Redes e Serviços*, pages 82–95. SBC.

Fraunhofer-Fokus and Tu-Berlin (2017). Open baton. `https://openbaton.github.io/`. Acessado em 08/04/2025.

Fulber-Garcia, V., Duarte Jr, E. P., Huff, A., and dos Santos, C. R. (2020a). Network service topology: Formalization, taxonomy and the custom specification model. *Computer Networks*, 178:107337.

Fulber-Garcia, V., Flauzino, J., Venâncio, G., Huff, A., and Junior, E. P. D. (2024). Breaking the limits: Bio-inspired sfc deployment across multiple domains, clouds and orchestrators. In *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6. IEEE.

Fulber-Garcia, V., Huff, A., Marcuzzo, L. d. C., Luizelli, M. C., Schaeffer-Filho, A. E., Granville, L. Z., dos Santos, C. R., and Junior, E. P. D. (2021). Customizable deployment of nfv services. *Journal of Network and Systems Management*, 29(3):36.

Fulber-Garcia, V., Marcuzzo, L., Huff, A., Bondan, L., Nobre, J., Schaeffer-Filho, A., dos Santos, C. R. P., Granville, L. Z., and Duarte, E. P. (2019). On the design of a flexible architecture for virtualized network function platforms. In *IEEE Global Communications Conference*, pages 1–6.

Fulber-Garcia, V., Souza, G. V. D., Jr, E. P. D., Tavares, T. N., Marcuzzo, L. D. C., Santos, C. R. D., Franco, M. F., Bondan, L., Granville, L. Z., Schaeffer-Filho, A. E.,

et al. (2020b). On the design and development of emulation platforms for nfv-based infrastructures. *International Journal of Grid and Utility Computing*, 11(2):230–242.

Hamdan, M., Mohammed, B., Humayun, U., Abdelaziz, A., Khan, S., Ali, M. A., Imran, M., and Marsono, M. N. (2020). Flow-aware elephant flow detection for software-defined networks. *IEEE Access*, 8:72585–72597.

Huff, A., Venâncio, G., Garcia, V. F., and Duarte, E. P. (2020). Building multi-domain service function chains based on multiple nfv orchestrators. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks*, pages 19–24. IEEE.

Maji, S., Veeraraghavan, M., Buchanan, M., Alali, F., Ros-Giral, J., and Commike, A. (2017). A high-speed cheetah flow identification network function (cfinf). In *IEEE Conference NFV-SDN*, pages 1–7.

Marcuzzo, L. C., Garcia, V. F., Cunha, V., Corujo, D., Barraca, J. P., Aguiar, R. L., Schaeffer-Filho, A. E., Granville, L. Z., and dos Santos, C. R. P. (2017). Click-on-osv: A platform for running click-based middleboxes. In *IFIP/IEEE IM*, pages 885–886.

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 *NSDI*, pages 459–473. USENIX.

OpenStack (2016). Tacker. `https://wiki.openstack.org/wiki/Tacker`. Acessado em 08/04/2025.

Peuster, M., Kampmeyer, J., and Karl, H. (2018). Containernet 2.0: A rapid prototyping platform for hybrid service function chains. In *2018 4th IEEE Conference on Network Softwarization and Workshops*, pages 335–337. IEEE.

Peuster, M., Karl, H., and Van Rossem, S. (2016). Medicine: Rapid prototyping of production-ready network services in multi-pop environments. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks*, pages 148–153. IEEE.

Quiles, F. R. (2024). Nfv-prime: Uma plataforma para prototipação e visualização de funções virtualizadas de rede. Master's thesis, PPGInf, UFPR.

Tavares, T. N., da Cruz Marcuzzo, L., Garcia, V. F., de Souza, G. V., Franco, M. F., Bondan, L., De Turck, F., Granville, L. Z., Junior, E. P. D., dos Santos, C. R. P., et al. (2018). Niep: Nfv infrastructure emulation platform. In *The 32nd IEEE AINA*, pages 173–180. IEEE.

Turchetti, R. C. and Duarte, E. P. (2015). Implementation of failure detector based on network function virtualization. In *2015 IEEE International Conference on Dependable Systems and Networks Workshops*, pages 19–25. IEEE.

Venâncio, G., Turchetti, R. C., and Duarte, E. P. (2019). Nfv-rbcast: Enabling the network to offer reliable and ordered broadcast services. In *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, pages 1–10. IEEE.

Venâncio, G., Turchetti, R. C., and Duarte Jr, E. P. (2022). Nfv-coin: Unleashing the power of in-network computing with virtualization technologies. *Journal of Internet Services and Applications*, 13(1):46–53.