GT-FENDE: Ecossistema Federado para Oferta, Distribuição e Execução de Funções Virtualizadas de Rede

RT3 - Avaliação dos resultados do protótipo

Leonardo da Cruz Marcuzzo, Muriel Figueredo Franco, Giovanni Venâncio de Souza, Cassiano Andrei Dias da Silveira Schneider, Lucas Bondan, Carlos Raniery Paula dos Santos

30/01/2018

1. Apresentação do protótipo desenvolvido

O protótipo desenvolvido foi elaborado de forma a refletir um cenário real de *marketplace* de funções de rede. Para tanto, sua arquitetura foi baseada nos elementos básicos que compõe o framework arquitetural de NFV definido pela ETSI. A arquitetura do protótipo está ilustrada em detalhes na Figura 1.

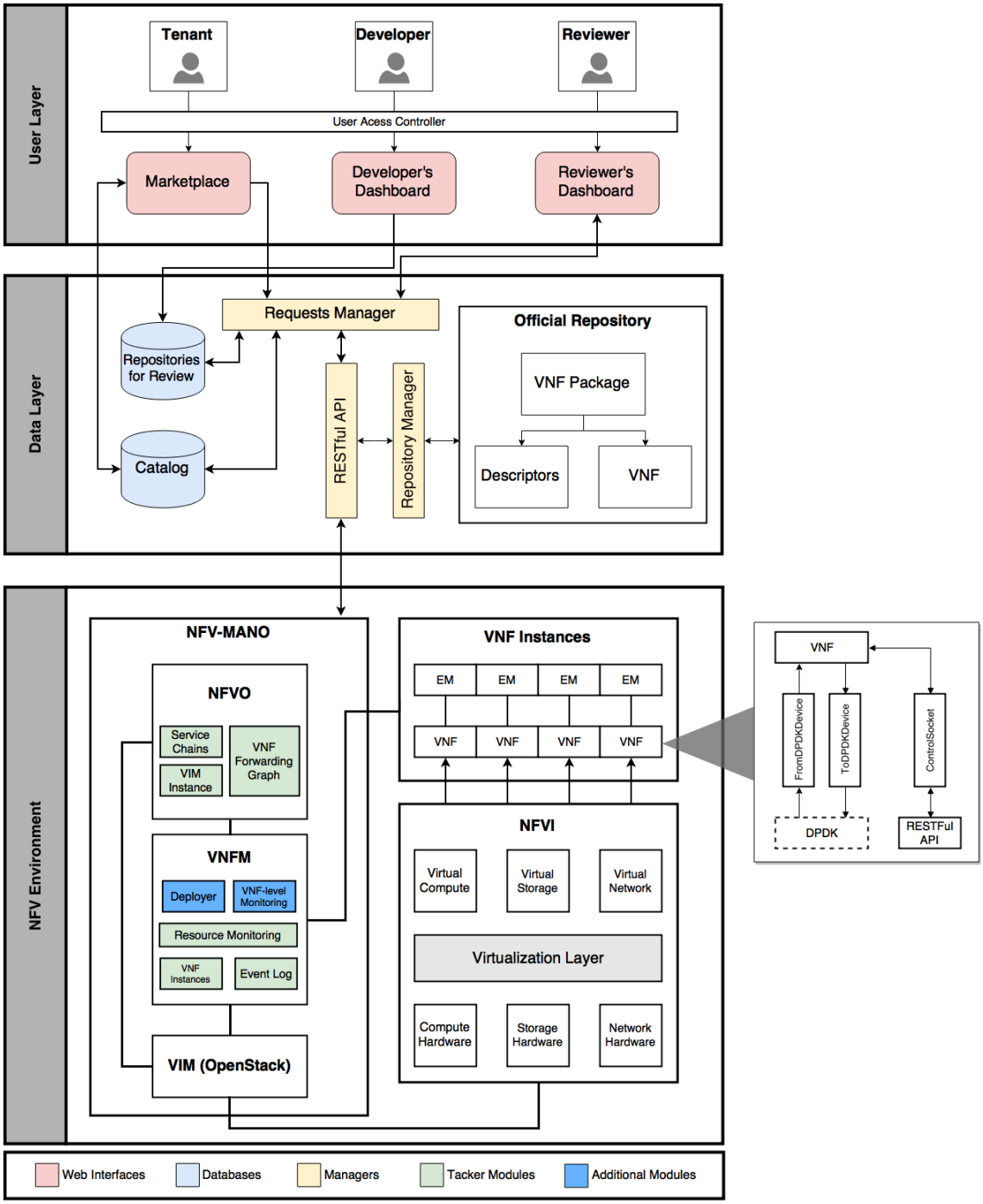


Figura 1: Arquitetura detalhada do protótipo da plataforma FENDE

A plataforma FENDE é dividida em três diferentes camadas, cada camada com módulos específicos para os diferentes níveis de operação da plataforma. As camadas do protótipo e seus módulos são descritos em detalhes a seguir.

1.1. Camada de Usuário (User Layer)

A camada de usuário contém os elementos responsáveis pela interação entre os diferentes atores com a plataforma. No protótipo implementado, foram considerados três diferentes atores: *(i)* Desenvolvedores *(Developers)*, os quais podem solicitar a inserção de suas VNFs na marketplace da plataforma*; (ii)* Revisores *(Reviewers),* responsáveis por analisar as solicitações dos desenvolvedores e adicionar ou remover VNFs dos catálogos; e *(iii)* Clientes (ou *Tenants*), os quais podem adquirir e implantar as VNFs disponibilizadas pelos desenvolvedores e aceitas pelos revisores.

Para gerenciamento e interação com os diferentes usuários, foi desenvolvido uma interface web utilizando Django Framework e Javascript. Cada uma das diferentes interfaces proveem recursos que possibilitam um conjunto de operações dentro do ecossistema. Os Clientes, por exemplo, possuem acesso apenas a interface denominada Marketplace, já os Desenvolvedores e Revisores possuem, respectivamente, acesso ao Painel de Desenvolvedores e ao Painel de Revisores.

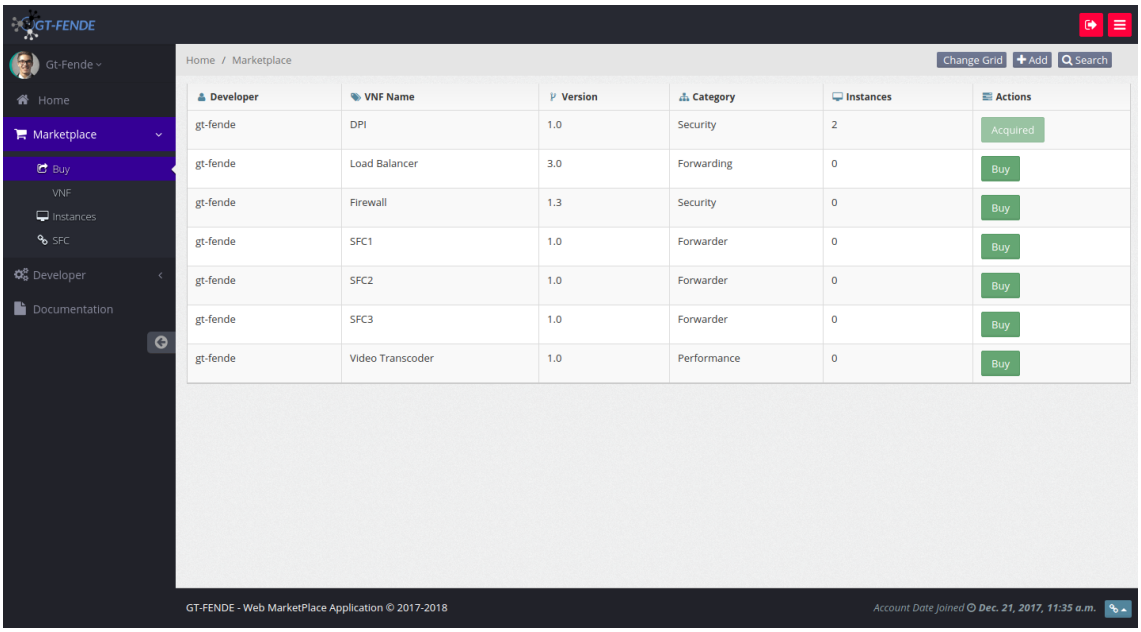


Figura 2: Interface da Marketplace

Na Marketplace (Figura 2), são apresentados as VNFs disponíveis no Catalog para serem contratadas. Nesta interface, os clientes podem contratar novos serviços e instancia-los. Além disso, é possível o gerenciamento e análise de estatísticas isoladas de cada uma das instâncias executadas. Ao navegar no menu lateral esquerdo, também é disponibilizada uma interface para que os usuários definam suas próprias cadeias de VNFs, também conhecidas como Service Function Chains (SFC)s. Por meio da criação de SFCs, é possível compor serviços mais complexos, como soluções de segurança compostas por diferentes funcionalidades, por exemplo, encadeando VNFs de firewall, *Deep Packet Inspection* (DPI), e *Intrusion Detection System* (IDS).

Já na interface do desenvolvedor, é possível realizar uma série de operações destinadas a submissão de novos repositórios, análise de repositórios submetidos e gerenciamento de repositórios ativos. Primeiramente, através da opção Publish, o desenvolvedor tem acesso a um formulário para submissão de um novo repositório para ser indexado pelo sistema. Após uma submissão ou atualização ser realizada, é possível acompanhar o andamento da revisão e comentários na tela de Submission Status (Figura 3).

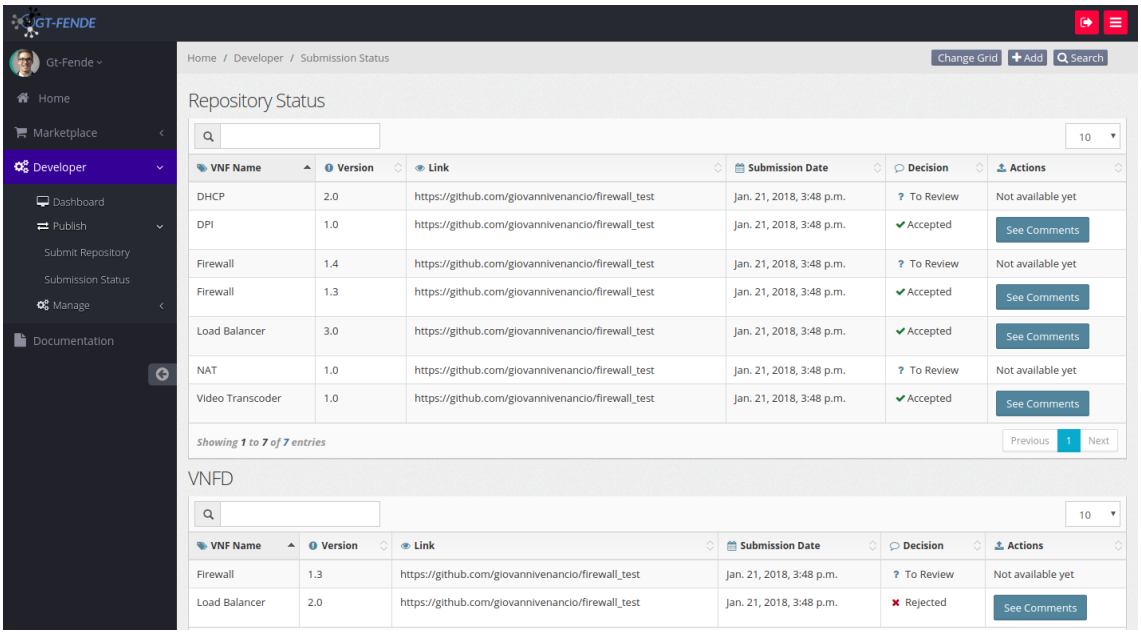


Figura 3: Status das Submissões de Repositórios do Desenvolvedor

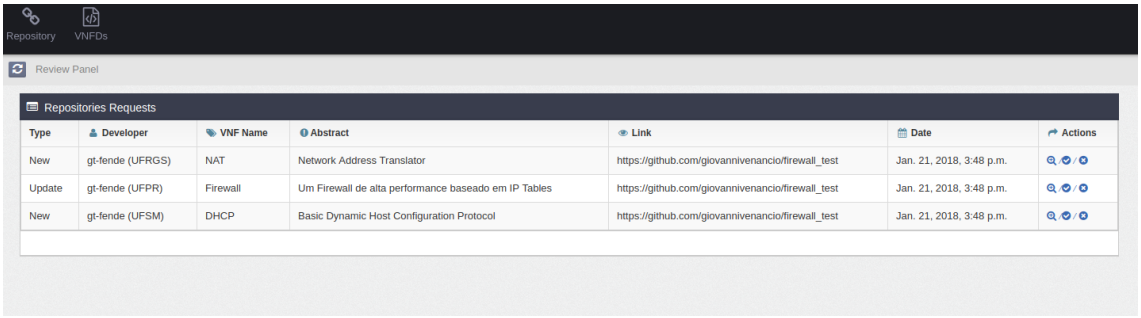


Figura 4: Painel de Revisões

No Painel de Revisão (Figura 4), é possível verificar todos os repositórios submetidos que podem ser aceitos ou rejeitados para integrar o catálogo do FENDE. Para isso, usuários com privilégios específicos podem acessá-lo, verificar as informações e os elementos presentes no repositório (e.g., código fonte, descritores e executável da VNF) e tomar uma decisão sobre a solicitação de indexação do repositório pelo FENDE. Portanto, nessa interface, o revisor pode consultar os detalhes de cada repositório submetido ao clicar no ícone de lupa e ter acesso a informações detalhadas que auxiliam no processo de convicção. Assim, caso o revisor se convença de que o repositório está estruturado da forma exigida pelo FENDE (e.g., descritores e arquivos de configuração) e contenha uma VNF funcional e satisfaz todos os requisitos, basta clicar no botão de aceite.

1.2. Camada de Dados (Data Layer)

Após as revisões e submissões de novos repositórios, uma série de eventos devem ocorrer no ecossistema para que os demais módulos possam utilizar as informações de forma sincronizada . Para isso, implementamos três módulos com papel de integrar a camada de usuários (User Layer) com o ambiente NFV (NFV environment). Esses módulos são descritos na camada de dados (Data Layer) e apresentados a seguir:

- **Requests Manager:** Responsável por armazenar as submissões de repositórios na tabela de Requests e também realizar a migração para a tabela Catalog quando o repositório é aceito.
- **RESTFul API:** Fornece um meio de comunicação entre os diferentes Managers. Suas principais funções são: I) Solicitar ao Repository Manager a criação ou atualização de um novo repositório e II) Solicitar ao Repository Manager e retornar o conteúdo dos descritores de uma VNF. Todos os módulos que precisarem de informações pertencentes ao Repository Manager podem encaminhar a requisição através da RESTFul API.
- **Repository Manager:** É o módulo responsável por criar, manter e gerenciar os arquivos e descritores das VNFs disponibilizadas no Catalog. Quando um repositório é aceito para fazer parte do FENDE, por exemplo, é papel do Repository Manager manter uma versão local desse repositório.

Um exemplo do funcionamento desta camada pode ser dar em função da interação ocorrida na interface apresentada na Figura 4. Ao clicar no botão de aceite, as seguintes tarefas são realizadas pelos módulos da camada de dados: Primeiramente, uma requisição via RESTful API é enviada para que o Repository Manager realize o clone do repositório de forma local. Após, o Request Manager se encarrega de deletar a entrada da tabela de Requests e criar uma nova entrada na tabela de Catalog apontando para o novo repositório criado. Por último, as políticas e informações contidas nos descritores da VNF indexada são extraídas e armazenadas no banco de dados.

1.3. Camada NFV (NFV Layer)

1.3.1. NFV MANO

Esta camada traz os principais elementos que compõe a arquitetura NFV proposta pela ETSI. Tais elementos são divididos em três subcamadas: *(i)* NFV MANO, onde encontram-se os elementos de gerenciamento de VNFs; *(ii)* VNF Instances, responsável pela execução das VNFs; e *(iii)* NFVI, a qual provê os recursos físicos necessários para a execução das VNFs. A seguir são descritos os módulos desenvolvidos para cada subcamada.

1.3.1.1. VNFM

O VNF Manager (VNFM) é um módulo do NFV-MANO responsável por gerenciar o ciclo de vida das VNFs. Em especial, o VNFM implementado consegue instanciar, remover e atualizar VNFs, além de criar composições entre elas (SFCs). A gerência das VNFs é feita em dois níveis: nível de hardware e nível de software. No nível de hardware são ajustados os recursos da máquina virtual que irá hospedar a função virtualizada. Estes recursos consistem de memória RAM, quantidade de CPU, quantidade de disco alocado, entre outros. No nível de software são configurados e instanciados na máquina virtual os softwares necessários para execução da função de rede.

Para a gerência no nível de hardware foi utilizado o plugin Tacker do OpenStack. O Tacker fornece uma API para a configuração básica de máquinas virtuais em um ambiente virtualizado. Para a gerência no nível de software é utilizado a API disponibilizada pelo Click-On-OSv. Além disso, foi desenvolvido um submódulo do VNFM responsável por consumir ambas as APIs e gerenciar de maneira completa o ciclo de vida das VNFs.

O VNFM recebe requisições da camada de usuário e executa os respectivos procedimentos no ambiente NFV. Em especial, o VNFM utiliza as APIs do Tacker e do Click-on-OSv nas suas funções. Todo o código foi desenvolvido em Python e todas as comunicações são feitas através de requisições REST. Além disso, todos os descritores mencionados a seguir (VNFD e VNFFGD) são descritos em JSON. A seguir é descrito como o VNFM desenvolvido gerencia o ciclo de vida das VNFs.

1. *Instanciação de uma VNF*: A operação de instanciação pode ser dividida em duas etapas: (i) criação e configuração da máquina virtual e (ii) execução da função

virtualizada. A primeira etapa consiste de dois passos, onde o primeiro adiciona um VNF Descriptor (VNFD) recebido da camada de dados para o catálogo de VNFs. O segundo passo é a criação da máquina virtual baseando-se no VNFD criado anteriormente. Uma vez que o tempo de criação de uma máquina virtual varia para cada sistema, o VNFM inicia um mecanismo de polling, onde ele verifica a cada intervalo de tempo se a máquina virtual está ativa. Quando o VNFM detecta que a criação da máquina virtual está pronta, inicia-se a segunda etapa. A segunda etapa consiste em inserir a função da VNF na máquina virtual, também recebida da camada de dados. Uma vez inserida, a função da VNF é executada e neste momento está totalmente funcional.

2. *Remoção de uma VNF*: O processo de remoção inicia-se quando o VNFM recebe um identificador único da VNF (vnf_id) e posteriormente remove a máquina virtual correspondente. Da mesma forma como na instanciação, um mecanismo de polling é iniciado para verificar, a cada intervalo de tempo, se a máquina virtual foi removida com sucesso. Por fim, o VNFD da VNF é removido do catálogo.

3. *Atualização de uma VNF*: A atualização de uma VNF pode ser dividida em dois níveis: (i) atualização de recursos e (ii) atualização da função virtualizada. No caso da atualização de recursos, a máquina virtual que hospeda a VNF pode atualizar, por exemplo, o número de CPUs, a quantidade de memória RAM e quantidade de disco alocados para a VNF. Essa atualização ocorre da seguinte forma. É recebido da camada usuário o vnf_id e o novo descritor da VNF, contendo as modificações que devem ser realizadas. Por exemplo, este descritor pode aumentar a quantidade de memória RAM de 512 MB para 1 GB ou alterar o número de CPUs de 8 para 4. O VNFM executa então a operação de atualização e, para concluir as alterações, a máquina virtual é reiniciada. Já para a atualização da função da VNF, o VNFM recebe o vnf_id e a nova implementação da função virtualizada. Da mesma forma como na instanciação, a nova função é inserida na máquina virtual. Após, a função antiga é removida e a nova função é executada.

4. *Criação de um SFC*: O Tacker além de fornecer a API básica para gerenciar as máquinas virtuais também disponibiliza uma interface para a criação de SFCs. O Tacker permite a criação de SFC através de um VNF Forwarding Graph Descriptor (VNFFGD), onde é possível especificar um encadeamento de VNFs. Entretanto, este descritor é complexo, uma vez que exige uma grande quantidade de informações da infraestrutura adjacente. A fim de facilitar a interação do usuário com o sistema e diminuir a complexidade para a criação do VNFFGD, criamos um descritor alto nível que abstrai o máximo de informações do sistema, diminuindo a quantidade de parâmetros que o usuário deve especificar. Neste descritor alto nível, o usuário deve preencher apenas três grupos de informação: (i) *Source*, com informações da origem do tráfego que irá entrar na SFC. podendo este ser tráfego externo (Internet) ou interno (outra VNF); (ii) *Path*, contendo informações das VNFs que compõe o SFC; e (ii) *ACL,* contendo uma lista de critérios pelos quais um pacote entra nesta SFC. Por exemplo, porta de destino.

É importante notar que todas estas informações estão disponíveis na própria Marketplace. Sendo assim, o processo de criação de SFCs ocorre da seguinte forma. Em primeiro lugar, o usuário cria e submete o descritor alto nível. O VNFM recebe este descritor e realiza um parser nestas informações. O parser extrai as informações do descritor e gera como saída o VNFFGD no formato esperado pelo Tacker, e o VNFM adiciona o VNFFGD criado no catálogo de SFCs. Após, o VNFM realiza a instanciação (utilizando a mesma função de instanciação descrita acima) de todas as VNFs especificadas no descritor e então inicia a SFC. Neste momento a SFC está totalmente funcional.

Apesar da utilização de um descritor alto nível aprimorar a interação do usuário com o sistema, existem ainda algumas melhorias que podem ser feitas para deixar a criação de SFCs mais dinâmica. Uma destas melhorias que ainda está em desenvolvimento é a criação de um formulário de SFCs. Neste formulário serão preenchidas as informações que antes eram inseridas manualmente no descritor alto nível. Esta abordagem possui várias vantagens, como por exemplo, a verificação em tempo real da inclusão de informações incorretas, o que diminui a chance de criar um SFC inconsistente

É importante destacar que foi desenvolvido também um mecanismo de rollback, que verifica para cada operação se ela foi concluída com sucesso. Em casos de erro, este mecanismo desfaz todas as modificações realizadas até o momento.

1.3.1.2. VIM

O protótipo da plataforma FENDE utiliza atualmente como Virtualized Infrastructure Manager (VIM) o OpenStack (versão stable/pike). A decisão por utilizar o OpenStack foi devido a sua vasta documentação, desempenho, e principalmente sua ampla adoção pela comunidade. Atualmente, o OpenStack tem sido adotado em inúmeras soluções de gerenciamento de infraestruturas no contexto de NFV, dada sua flexibilidade para implementação de módulos de gerenciamento para controle da infraestrutura, como é o caso do VNFM implementado na plataforma FENDE, cujo protótipo utiliza o módulo Tacker do OpenStack.

1.3.1.3. VNF Instances

Máquinas virtuais e containers responsáveis por executar as VNFs são denominados VNF Instances. Cada instância é responsável apenas por uma função específica (*e.g.,* firewall, roteador, NAT) que podem ser encadeadas através de SFCs para criar serviços mais complexos. Atualmente, o projeto possui suporte a VNFs baseadas no Click Modular Router, que por sua vez são executadas no VNF Server. No entanto, pela portabilidade do Click, extensões para execução de VNFs em containers e máquinas virtuais Linux podem ser desenvolvidas.

1.3.1.3.1. VNF Server: Click-on-OSv

Com o objetivo de criar uma plataforma específica para a composição e execução de VNFs, a plataforma Click-on-OSv foi desenvolvida. Esta plataforma é composta de três componentes básicos: *(i)* OSv, um sistema operacional minimalista (*i.e.,* Unikernel) capaz de executar em infraestruturas virtualizadas com uma interface REST integrada e extensível; *(ii)* DPDK, um acelerador de processamento de pacotes; e *(iii)* Click Modular Router, um roteador virtual capaz de compor diferentes tipos de funções de rede.

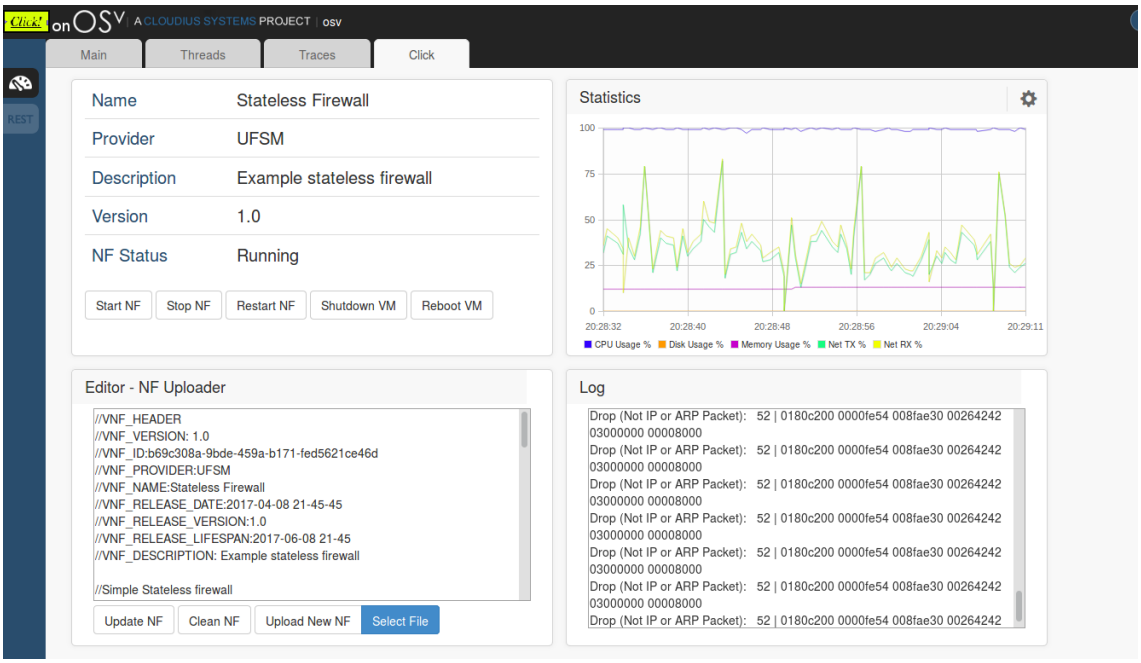
O desenvolvimento da plataforma corresponde a integração destas três ferramentas, resultando em uma plataforma que atende os requisitos especificados pelo ETSI. O sistema operacional OSv apresenta uma arquitetura minimalista, sendo considerado um Unikernel, ou seja, um sistema operacional cujo propósito é a execução de um único software. Assim, diferente de máquinas virtuais baseadas em sistemas tradicionais como o Linux, o OSv possui uma utilização de memória e disco reduzidos, na ordem de megabytes. Isto está de acordo com a especificação do ETSI, dado que VNFs devem ter custos computacionais reduzidos, aumentando a quantidade de VNFs que podem ser executadas em um servidor de virtualização, e serem capazes de serem migradas. Além disso, o OSv possui uma API REST e uma interface web que podem facilmente ser extendidas. Ambas foram extendidas para a utilização no VNF Server, sendo que a API REST é utilizada para comunicação com o VNFM enquanto que a interface web fornece uma página (Figura 5) para que a VNF seja configurada manualmente por um usuário.

Figura 5: Interface web do VNF Server (Click-on-OSv)

O acelerador de pacotes DPDK utiliza conceitos como processamento de pacotes em lotes e polling para ter um alto desempenho em infraestruturas virtualizadas. Embora o DPDK possua uma API capaz de criar funções avançadas de rede, no contexto do VNF Server ele é utilizado para encaminhar os pacotes que chegam a VNF para o Click Modular Router.

Já a execução das funções de rede é feita pelo Click Modular Router. O Click permite que funções sejam compostas por pequenos componentes denominados elementos. Cada elemento corresponde a uma parte específica de uma função, como um classificador, marcador de pacotes e entrada/saída de interfaces de rede. Como o Click já possui diversos elementos implementados, possui bastante flexibilidade, trabalhando com frames Ethernet, IPv4 e IPv6, e Wifi.

A implementação da plataforma consistiu de adaptações no Click, DPDK e extensão da API REST e interface web do OSv. No Click, foram feitas modificações no processo de compilação, onde o binário passou a ser compilado como uma biblioteca compartilhada (shared library) com PIC (position-independent code), dado que o Osv não suporta apenas este tipo de



executável, por não dar garantias sobre o gerenciamento de memória, e no processo de linkagem de bibliotecas. Já o DPDK, utilizou-se uma versão adaptada para funcionamento com o OSv, com modificações no controle de interrupções e no suporte a drivers paravirtualizados do tipo virtio. Por já existir esta versão modificada que funcionava com o OSv, foram feitas apenas adaptações para que o Click a reconhecesse como uma compilação válida e fosse linkada sem problemas. Por fim, no OSv a API REST foi estendida com a seção “click_plugin”, apresentando as seguintes funções:

- */click_plugin/version*: Retorna a versão do Click que está sendo executada
- */click_plugin/running*: Retorna se a VNF está ou não em execução;
- */click_plugin/start*: Inicia uma VNF;
- */click_plugin/stop*: Para uma VNF;
- */click_plugin/read_file*: Retorna a função que atualmente está carregada na VNF;
- */click_plugin/write_file*: Passando uma nova função como parâmetro, atualiza a VNF;
- */click_plugin/vnf_identification*: Retorna informações sobre a função atualmente carregada (nome, autor, descrição, versão e unique_id);
- */click_plugin/metrics*: Retorna estatísticas de uso de CPU, memória, disco e rede;
- */click_plugin/log*: Retorna o log de execução da VNF
- */click_plugin/custom_request*: Envia uma requisição customizada ao Click (ex contador de pacotes do elemento de entrada)

Também foram desenvolvidas funções de rede para exemplificar a criação e o funcionamento do Click Modular Router. Estas funções, junto com uma explicação mais detalhada sobre o desenvolvimento delas e de novas funções pode ser visto no relatório “Implementação de funções virtualizadas de rede utilizando o Click Modular Router”

1.3.1.4. NFVI

Para a execução das VNF Instances é necessário um servidor de virtualização com um hypervisor compatível. De acordo com o ETSI, estes servidores formam o bloco funcional NFV Infrastructure. Dentro do projeto, dois servidores são utilizados para executar as instâncias, enquanto o Marketplace e o VNFM é executado em um servidor separado. Além disso, servidores de virtualização dos projetos FIBRE e FUTEBOL também podem ser utilizados para executar as instâncias. Todos os servidores são conectados entre si através de uma VPN, de modo que a comunicação seja possível e as instâncias executem em uma mesma subrede, permitindo a utilização de SFCs sobre múltiplos servidores.

1.3.1.4.1. Servidores do projeto (distribuídos)

Os servidores do projeto estão configurados com uma instalação do OpenStack, versão stable/pike, com o hypervisor KVM. Dentro destes servidores, instâncias do VNF Server podem ser configuradas e instaladas a partir do VNFM, que por sua vez é acessado pelos usuários através do marketplace. Também, em cada um dos servidores existe um agente responsável por coletar estatísticas das VNFs e enviar a um banco de dados temporal (timeseries database), assim podendo verificar-se estatísticas sobre uso de cada um dos servidores e suas respectivas VNFs. Os servidores são conectados entre si através de uma VPN, de modo que, mesmo que duas VNFs estejam executando em servidores diferentes, estas podem comunicar-se entre si.

1.3.1.4.2. Servidores FIBRE/FUTEBOL (lembrar de falar do driver de comunicação)

Os servidores de virtualização, tanto do FIBRE como do FUTEBOL, não utilizam o OpenStack. No caso do FIBRE, é utilizado a versão 4.1 do hypervisor Xen, enquanto que o FUTEBOL utiliza o KVM. Desta forma, foi necessária a implementação de um wrapper para o VNFM funcionar nestas duas infraestruturas. Assim, para a infraestrutura do FIBRE, foram criadas chamadas para o backend Xend, de forma que é possível criar e deletar instâncias, a partir do parse do VNFD original.[não foi testado]

Pelo FUTEBOL utilizar KVM e ter suporte ao libvirt, não houve problemas quanto a instanciação do VNF Server neste ambiente. Foi utilizada a biblioteca em python do libvirt, de modo que instâncias podem ser criadas, configuradas, e apagadas. A função para inicializar a VNF e reconfigurá-la é a mesma do VNFM, pois sua utilização depende apenas do VNF Server e do IP da instância.

As duas infraestruturas serão conectadas da mesma forma a VPN dos servidores que fazem parte do projeto, de modo que a maior diferença entre estas infraestruturas e o servidores do projeto é a utilização de drivers de comunicação.

2. Resultados dos testes

2.1 VNF Server

Com o objetivo de avaliar o desempenho do VNF Server, foram realizador testes de tempo de boot e vazão. Estas métricas são importantes de serem avaliadas em um ambiente NFV, pois tempo de boot é utilizado para verificar o tempo de instânciação e recuperação de VNFs, enquanto que a vazão apresenta o limite de processamento das VNFs.

Assim, para o tempo de boot foram utilizados diversas configurações e sistemas operacionals, com os hypervisors Virtual Box e KVM, de modo a testar a compatibilidade do VNF Server em ambientes Windows, conforme Tabela 1, enquanto que os testes de vazão foram realizados em um servidor Intel Xeon E5620@2.40 com 8 núcleos e 12 GB de memória DDR3, em um sistema operacional Debian 8 e hypervisors Xen 4.4.1 e KVM 1.2. Nos testes de vazão, optou-se por comparar a plataforma desenvolvida com a plataforma ClickOS, de propósito semelhante e com boa aceitação na comunidade acadêmica.

Sistema Operacional	Hardware	Hypervisor	Tempo de Boot
Debian 8	Intel Core i5-5200U @ 2.20 GHZ 8GB DDR3	Virtual Box 4.3.40	962,2 ms
		KVM	3570 ms
Ubuntu 14.04	Intel Core i3-4010U @ 1.70Ghz	Virtual Box 4.3.40	852,8 ms

	8GB DDR3	KVM	3771,8 ms
Windows 10	Intel Core i3-4010U @ 1.70Ghz 8GB DDR3	Virtual Box 4.3.40	11655,2 ms

Tabela 1: Tempo de boot do VNF Server em diversas infraestruturas

Nota-se que, em testes onde o KVM é utilizado como hypervisor, o tempo é maior devido a VNF ter que esperar receber um IP da rede por DHCP, enquanto que no Virtual Box não há espera por se tratar de uma rede interna do hypervisor. A única exceção é o tempo de boot no Windows 10, onde o Virtual Box 4.3.40 não é oficialmente suportado. Assim, assume-se uma média de tempo de instanciação inferior a 1 segundo, desconsiderando o tempo de DHCP.

Já a Figura 6 apresenta a vazão da plataforma em comparação com o ClickOS para diversos tamanho de pacotes. É possível ver que a plataforma possui desempenho superior em todos os casos a partir de pacotes de 256B, enquanto seu desempenho e ligeiramente pior para pacotes entre 64 e 128B.

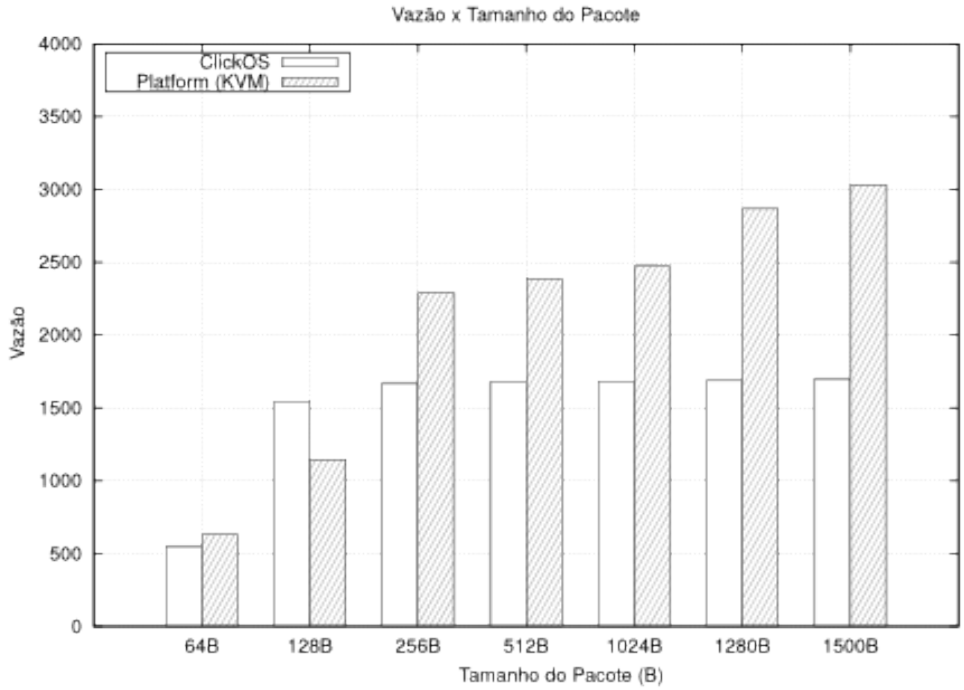


Figura 6: Vazão do VNF Server para diferentes tipos de pacotes

2.2 VNF Manager

Para execução dos testes de desempenho do VNFM foi utilizado um servidor Dell PowerEdge com um processador Intel Xeon E3-1220v6 @3.00GHz, 4 núcleos e 8 GB de memória, executando em um Ubuntu 16.04. Para avaliar as operações de gerência do ciclo de vida das VNFs, foi utilizado um protótipo do VNFM que possui as seguintes funções: Create, Delete, Update e Update Function.

Cada função do VNFM foi executada da seguinte forma. A função Create foi executada instanciando uma VNF com um firewall, e recursos de 512 MB de RAM, 1 CPU e 1 GB de disco. A função Update foi executada alterando a memória RAM alocada para a VNF de 512 MB para 1 GB, enquanto os demais recursos permaneceram iguais. Para a função Update Function foi alterada a função em execução para um VNF forwarder. Por fim, a VNF foi removida utilizando a função Delete. Todos os testes foram realizados 30 vezes, atingindo um intervalo de confiança de 95%.

O primeiro experimento tem como objetivo medir o tempo de execução de cada uma das operações. Além disso, foi calculado o tempo de cada suboperação que compõe cada uma das funções. Como pode ser observado na Figura 3, a operação Create foi a operação com maior duração, sendo que a sub operação de polling consome aproximadamente 90% do tempo total de execução da operação, conforme mostra a Figura 4. Isto ocorre pois o polling deve verificar periodicamente e aguardar a infraestrutura terminar o processo de criação da VM, para que a função de rede possa ser instanciada. Já a operação Update Function precisa realizar o upload de uma nova função de rede e aguardar a reinicialização da VNF. Por fim, a operação Update após atualizar a descrição da VM aguarda a reinicialização da VM, enquanto a operação Delete envia comandos para que a VNF seja completamente removida.

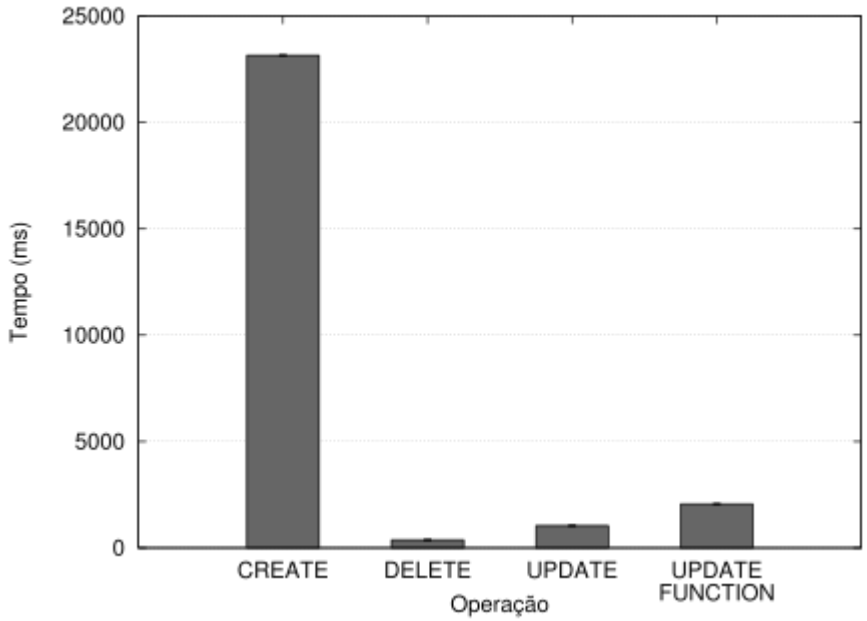


Figura 3. Tempo total de execução das operações.

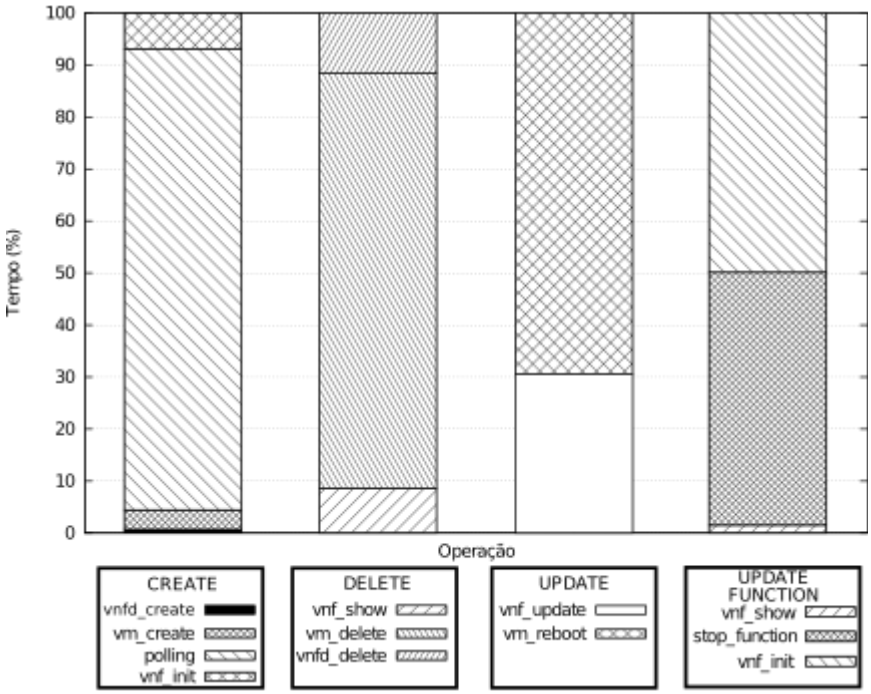


Figura 4. Tempo de execução das sub operações em porcentagem.

O segundo experimento tem como objetivo medir a quantidade de recursos utilizado pelo VNFM. A Figura 5 apresenta a utilização de memória e CPU de cada operação. Com relação a memória RAM, a memória utilizada é ocupada pelas bibliotecas utilizadas para o desenvolvimento do VNFM. Como estas bibliotecas são utilizadas por todas as operações, o consumo de memória é semelhante. Já para a utilização de CPU, é possível perceber que a operação Delete apresenta a maior utilização de CPU, enquanto que as operações Create e Update Function, mesmo possuindo suboperações mais longas, apresentam um menor consumo de CPU. Isto deve-se ao fato de que, no caso da operação Create, apesar da sub operação de polling ser a mais longa, ela apenas aguarda a VM ser criada, não realizando nenhum processamento neste intervalo. Já a operação Update Function precisa aguardar a VNF reinicializar para finalizar. Desta forma, como a operação Delete não precisa aguardar nenhum tipo de resposta, durante todo o tempo da sua execução ela está realizando algum tipo de processamento. Por fim, a operação Update não apresenta um consumo de CPU significativo, já que ela apenas atualiza o VNFD e aguarda a reinicialização.

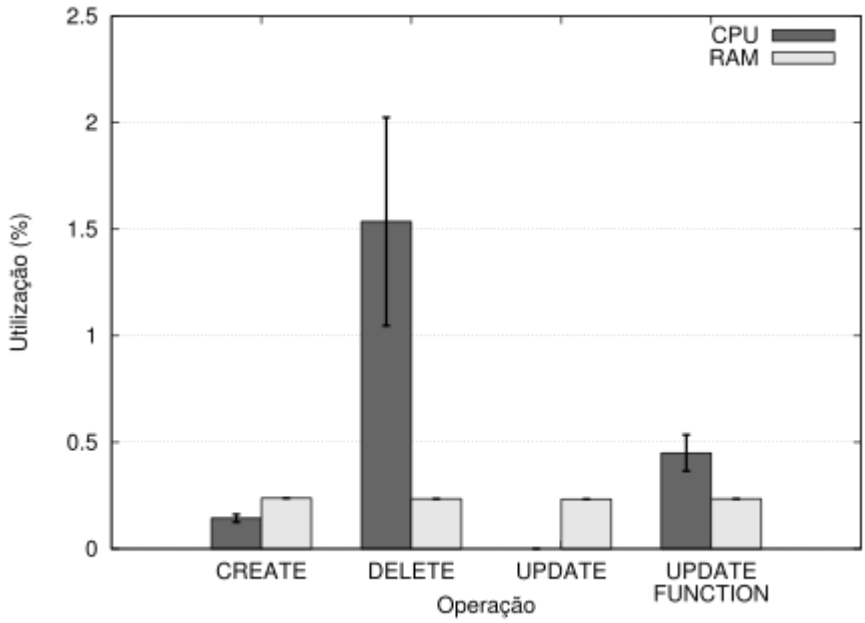


Figura 5. Uso de CPU e memória durante as operações.

Sendo assim, o maior consumo dentre todas as operações foi de apenas 1.5% de uso da CPU, enquanto que o consumo de memória RAM corresponde a 0.25% da memória disponível. Dessa forma, pode-se concluir que o impacto do VNFM nos recursos do servidor é mínimo.

3. Avaliação

A plataforma FENDE possibilita a distribuição controlada de VNFs em diferentes ambientes, sejam eles de produção ou experimentais. Por meio do FENDE, desenvolvedores podem disponibilizar suas soluções de maneira segura garantida à operadores de rede com diferentes interesses. Da mesma forma, operadores de rede interessados em se valer das vantagens oferecidas pela aplicação de VNFs em seus ambientes de rede, seja para redução de custos ou mesmo para flexibilizar o provisionamento de serviços aos seus usuários.

Com o FENDE se tornando um serviço RNP, tanto universidades quanto companhias do ramo de rede poderão se cadastrar na plataforma, a fim de desenvolver, instanciar e testar novos serviços de rede. Com a flexibilidade proporcionada pelo FENDE, pode-se reduzir o tempo-para-mercado (ou time-to-market - tempo entre a análise de um produto e sua disponibilização), uma das principais vantagens na adoção de NFV.

Além disso, outra possibilidade será aplicar diferentes formas de monetização (ou *billing*) na utilização da plataforma. Por exemplo, uma vez que a VNF de determinado desenvolvedor é adquirida por um cliente na plataforma, o mesmo pode pagar pelo “aluguel” da função, de acordo com a utilização da mesma. Dessa forma, parte da renda gerada pela utilização das VNFs ficaria com o desenvolvedor e outra com o FENDE pelo gerenciamento da plataforma.