PyCOO: Uma API em Python para Plataforma Click-On-OSv

Vinícius Fülber Garcia[1], Thales Nicolai Tavares[1], Leonardo da Cruz Marcuzzo[1], Lucas Bondan[2], Muriel Figueredo Franco[2], Giovanni Venâncio de Souza[3], Alberto Egon Schaeffer-Filho[2], Carlos Raniery Paula dos Santos[1]

[1]Departamento de Computação Aplicada – Universidade Federal de Santa Maria (UFSM)
Av. Roraima nº 1000 – 97.105-900 – Santa Maria – RS – Brazil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[3]Departamento de Informática –Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba, PR – Brazill

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

{lbondan,mffranco,alberto}@inf.ufrgs.br, gvs11ufpr@gmail.com

Abstract. *The NFV technology is an option to keep a network function infrastructure in a more flexible, facilitated management and reduced costs way. Specialized platforms for VNFs execution, such Click-On-OSv, uses packet accelerators to recover part of the lost performance. This work presents PyCOO, an API developed in Python that enables the control from infrastructure to Click-On-OSv's internal routines to facilitate remote utilization and prototyping environments creation.*

Resumo. *A tecnologia NFV é uma opção para manter uma infraestrutura de funções de rede de forma mais flexível, com gerência facilitada e custos reduzidos. Plataformas especializadas na execução de VNFs utilizam aceleradores de pacotes para recuperar parte desse desempenho perdido, como é o caso do Click-On-OSv. Este trabalho apresenta o PyCOO, uma API desenvolvida em Python que possibilita o controle desde da infraestrutura até as rotinas internas do Click-On-OSv a fim de facilitar a utilização remota da mesma e criação de ambientes de prototipação.*

1. Introdução

Funções Virtualizadas de Rede (NFV) se tornaram um paradigma que objetiva desacoplar as funções de rede de seu hardware dedicado. Para que isso seja possível sistemas de virtualização já existentes são utilizados (*i.e., hypervisor*) para instanciar servidores virtuais, estes servidores rodam em *hardware* genérico [ETSI 2012], o que contribuí para reduzir o *time to market* e custos de capital e operacionais (CAPEX e OPEX) [Mechtri et al. 2017].

Um conjunto ordenado de funções virtualizadas de rede e elementos que definem rotas pelas quais pacotes e fluxos definem uma Cadeia de Funções de Serviço (SFC) [Quinn and Nadeau 2015]. Do ponto de vista operacional uma SFC pode ser descrita através de um grafo chamado de Grafo de Encaminhamento (VNF-FG)[ETSI 2013], este

grafo contém nós representando entidades como VNFs e arestas como conexões lógicas. Pontos de entrada e saída de dados também são definidos em um VNF-FG e servem para indicar o inicio e o fim da SFC.

Apesar do paradigma NFV apresentar diversas características vantajosas em relação a utilização de *middleboxes*, como facilidade para realização de operações de escala e maior liberdade de desenvolvimento, o uso de *hardware* genérico para a execução de funções de rede apresenta desempenho inferior quando comparada ao uso de *hardware* dedicado [Hwang et al. 2015]. Esforços para reduzir essa diferença de desempenho são realizados através do desenvolvimento de plataformas de software específicas para hospedar e executar funções de rede virtualizadas.

Plataformas para execução de NFV como ClickOS [Martins et al. 2014], OpenNetVM [Zhang et al. 2016] e Click-On-OSv [Marcuzzo et al. 2017], utilizam tecnologias de aceleração de pacotes Netmap [Rizzo 2012] ou Intel DPDK [Intel 2017]. O desenvolvimento e otimizações das plataformas é uma etapa fundamental para a disseminação e popularização do paradigma.

Diversas soluções para controle de infraestrutura estão disponíveis [Baton 2017][Sefraoui et al. 2012][Tacker 2017]. Essas aplicações têm capacidade de realizar a instanciação e manter a infraestrutura de máquinas virtuais que hospedam funções de rede virtualizadas. Entretanto, essas ferramentas não são capazes acessar as plataformas internamente para manipularem as funções a serem executadas, para isso sistemas auxiliares que se adaptam as peculiaridades de cada plataforma são desenvolvidos.

Este trabalho apresenta uma API desenvolvida em Python para acesso e controle de VNFs que executam sobre a plataforma Click-On-OSv. A API consiste de dois grandes módulos (infraestrutura e funções virtualizadas). O módulo de infraestrutura tem como responsabilidade o gerenciamento de Máquinas Virtuais (VMs) enquanto que o módulo de funções virtualizadas é responsável pelo gerenciamento de VNFs, SFCs e pela aplicação de *Scripts* de Execução.

A API elaborada pode ser utilizada para diversos tipos de aplicações que necessitem instanciar e controlar VNFs. A partir do conjunto de funções dispostas, é possível criar ambientes de prototipação da plataforma Click-On-OSv semelhantes ao ESCAPE [Csoma et al. 2014]. Também, a solução pode ser utilizada para a criação de topologias de rede de maneira automática para a realização de testes, como comparações de desempenho de diferentes implementações de uma mesma função em Click.

O restante desse artigo está organizado como se segue: a seção 2 expões as características de acesso e controle das principais ferramentas e plataformas para execução de NFV. A seção 3 trás em detalhes os requisitos e o modo de funcionamento da API desenvolvida. A seção 4 apresenta a conclusão do trabalho.

2. Plataformas e Interfaces de Acesso e Controle

Diferentes plataformas para a execução de VNFs estão disponíveis. Essas plataformas apresentam diferentes meios para gerenciamento das funções as quais se destinam a executar. A simplificação das interfaces de gerência das funções virtualizadas é um passo fundamental para a popularização da tecnologia NFV.

O Click Modular Router [Kohler et al. 2000] utiliza uma arquitetura flexível e modular de software para criar e encaminhar pacotes. Sua arquitetura baseia-se na utilização de elementos específicos para processamento de pacotes, com funcionalidades específicas (*e.g.,* entrada de pacotes, classificadores L2 e L3). Estes elementos podem ser interconectados para a criação de novas funcionalidades complexas (*e.g., Switches*, Roteadores).

Cada elemento implementa uma função simples, a qual possui um ponto de entrada e saída e pode ser configurado individualmente, através de *handlers* internos. O gerenciamento de uma função de rede executando no Click é feita através de um elemento chamado `ControlSocket`. Este elemento expõe, através de um socket TCP, os *handlers* de cada um dos elementos que compõem a função, que por sua vez podem ser configurados individualmente através de um agente externo. Alguns dos *handlers*, como contadores, podem ser utilizados para coleta de métricas de desempenho da função.

O ClickOS [Martins et al. 2014] é uma plataforma desenvolvida para a execução de VNFs por meio de de virtualização utilizando o Mini-OS com drivers paravirtualizados modificados para suportar o *netmap*, este último é uma solução para realização de entrada e saída de pacotes em alta velocidade. Em relação a criação e execução das VNFs, a solução Click Modular Router é utilizada, este executa as funções de rede. Sua arquitetura têm como foco o flexibilidade, desempenho e escalabilidade.

A gerência de uma instância do ClickOS é realizada através da modificação de variáveis internas utilizando o XenStore. Esta interface permite apenas o controle do ciclo de vida da instância e o *upload* de funções, de forma que os *handlers* internos do Click não podem ser configurados (*i.e.,* não há um `ControlSocket` em execução). Assim, não é possível coletar métricas internas, e alterações em elementos implica no *upload* de uma nova função de rede.

A plataforma OpenNetVM [Zhang et al. 2016] emprega o uso de *containers* Docker para a execução de VNFs em servidores em conjunto com o *framework* de aceleração de pacotes DPDK, o qual objetiva minimizar a sobrecarga do processamento de pacotes e fornecer redes com grandes taxas de transferência e baixa latência em ambientes virtualizados [Ramakrishnan 2016]. A arquitetura desta plataforma é composta de dois módulos principais: *NF Manager* e as NFs. O *NF Manager* é responsável por manter o estado da rede através da gerência de NFs e roteamento dos pacotes. Já as NFs se comunicam com o *NF Manager* através da *NFLib*, informando quando estão prontas para serem executadas.

Nota-se que as NFs do OpenNetVM não são implementadas utilizando o Click, e devem sem implementadas em linguagem C, utilizando *callbacks* da *NFLib* para definir pontos de entrada, processamento e saída. Assim, não existem elementos prontos que podem ser utilizados para a construção de uma NF, bem como não existe uma forma pré-definida de recuperar métricas e reconfigurar NFs pois esses devem ser definidos durante a implementação das mesmas.

A plataforma Click-On-OSv (COO) utiliza o sistema operacional minimalista OSv [OSv 2017] executando o Click Modular Router[Kohler et al. 2000] e o acelerador de pacotes Intel DPDK para desempenhar funções virtualizadas de rede. O Click-On-OSv se diferencia de seus similares por prover tanto uma interface gráfica quanto uma interface REST de forma nativa para gerenciamento de VNFs.

Para a gerência do Click-On-OSv, é possível utilizar tanto requisições REST como

uma interface Web. A interface REST é implementada como uma extensão da interface nativa do OSv, e permite controle do ciclo de vida, tanto da função virtualizada como da instância da máquina virtual. Internamente, um `ControlSocket` conecta a instância do Click com a interface REST do OSv, o que permite que métricas internas das funções possam ser coletadas externamente através da interface do OSv. A interface Web, por sua vez, é uma interface gráfica para os métodos REST, o que permite que as métricas coletadas possam ser exibidas através de gráfico, com botões para controle do ciclo de vida, propriedades dos descritores das VNFs. A plataforma também implementa um editor de funções e um *log* de erros, que podem ser utilizados para o tratamento de eventuais problemas que ocorram na instanciação da função.

3. PyCOO

A API para acesso e controle de VNFs foi desenvolvida com o intuito de facilitar o gerenciamento da plataforma Click-On-OSv e facilitar a realização de tarefas através da definição de comandos de alto nível compostos por um conjunto de comandos básicos. Além disso, por se tratar de um grupo de funções que podem se importadas, a API como um todo ou apenas módulos dela podem ser utilizados como parte da implementação de softwares terceiros.

A API foi desenvolvida em Python, utiliza o *hypervisor* KVM para oferecer a infraestrutura de instanciação necessária, Linux *Bridges* para prover conexões entre VNFs para a formação da SFC e também com elementos externos a mesma, além de fazer uso de chamadas REST para executar ações internas a plataforma. Toda a comunicação entre usuário e API pode ser realizada através arquivos de configuração em formato JSON.

A Figura 1 apresenta a estrutura da API, os elementos em azul representam os módulos implementados como parte deste trabalho, em vermelho são apresentados agentes previamente existentes necessários para a utilização da API, em verde são apresentados os arquivos de configuração que definem configurações das VNFs, SFCs e as ações a disponibilizadas que são consumidos pela API.

A API [1] é dividida em quatro módulos, de VMs, de VNFs, de SFCs e de *Scripts* de Execução. Cada módulo é acoplado e utiliza dados providos por um terceiro, a exceção do controle de VMs que é independente e substituível. Uma instância virtual executando o sistema Click-On-OSv é caracterizada como uma VNF, um conjunto de VNFs e informações de rotas entre elas designa uma SFC, por fim, *scripts* de execução indicam sequências de ações a serem tomadas, também podem ser definidas situações de exceção.

3.1. Gerenciamento de VMs

A API utiliza uma imagem padrão da plataforma Click-On-OSv, todas as VNFs apresentam um descritor simples que altera o arquivo de instanciação usado pelo KVM através da CLI Virsh, os dados contemplados pelo descritor são:

- ID: identificador único, será o nome dado ao diretório contendo os arquivos da VNF e imagem da VM;
- *Memory*: quantidade dedicada de memória RAM para a VM que será instanciada;
- VCPU: número de CPUs virtuais dedicadas a execução da VM;

[1] Código e exemplos: https://github.com/ViniGarcia/ClickOnOSvManager

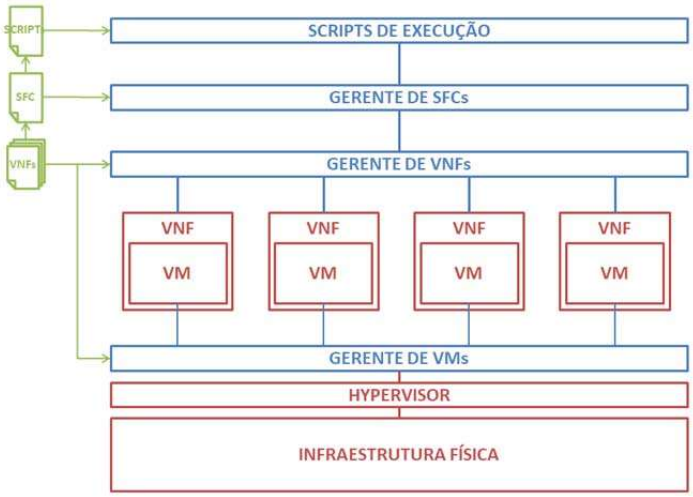

Figura 1. Elementos e Relações Referentes a API

- *Management MAC*: Endereço MAC dedicado para a interface de gerência da plataforma;
- *Interfaces*: cada interface contém um ID que indica a Linux *Bridge* a ser utilizada e um MAC dedicado a ela, essa informação é usada somente quando a instanciação for de uma VNF isolada e não de uma SFC.

As VMs são criadas em modo temporário quando iniciadas, ou seja, seu ciclo de vida começa quando são instanciadas e termina quando são desligadas. Após a configuração da VM ser lida, um objeto do tipo VNF é criado, sendo disponibilizo métodos de controle da VM, como ligar e desligar.

Métodos para mudança de configuração de VNF e obtenção de informações da VM também são disponibilizados. Finalmente, métodos para a criação do diretório da VNF e cópia de seus subsequentes arquivos e para a destruição dos mesmos são oferecidos para possibilitar o reuso de IDs e economia de memória.

3.2. Gerenciamento de VNFs

Consiste no gerenciamento das funções que a plataforma Click-On-OSv é capaz de realizar, é feito através da interface REST disponibilizada pela plataforma acessada via requisições HTTP enviadas através da API. Entre as ações que podem ser realizadas estão:

- *POST*
  - *Start*: ativa o DPDK e o Click, após executa a função que está reservada dentro da plataforma;
  - *Stop*: para a função Click que esta executando na plataforma;
  - *Function*: ao enviar o caminho do arquivo que contém uma função Click, a mesma é repassada a plataforma e passa a ser a função reservada.
- GET
  - *Running*: retorna informação do estado do módulo Click, se o mesmo esta ativo ou não;

– *Function*: retorna a função de rede reservada nas configurações internas da plataforma;
– *Identification*: retorna os metadados contidos na especificação da função de rede reservada;
– *Metrics*: retorna métricas sobre a execução da função de rede na plataforma;
– *Log*: retorna informações de *log* sobre inicialização, execução e conclusão de uma função de rede.

3.3. Gerenciamento de SFCs

O gerenciador de SFCs utiliza um descritor que apresenta dados básicos de VNFs a serem utilizadas e conexões entre as mesmas. Antes de ser submetida, todos os dados presentes no documento de descrição são submetidos a uma análise de integridade quanto a existência das VNFs requisitadas e do grafo gerado. As informações necessárias para a definição de uma SFC são:

- ID: identificador único, será o nome pelo qual o programa identificará a SFC quando em execução;
- VNFs: conjunto identificando todas as VNFs que fazem parte da SFC. É necessário que o arquivo de configuração das VNFs escolhidas esteja presente no sistema;
- *Incoming Point* (IP): ponto de entrada de dados. Existe apenas um para cada SFC, recebe dados de uma fonte externa a SFC e repassa a primeira VNF do sistema, é definido por um ID e uma ponte;
- *Outcoming Points* (OPs): pontos de saída de dados, ou seja, pontos finais onde dados são transmitidos de uma VNF para um destino externo a SFC. Como o IP, é definido por um ID e uma ponte;
- *Connections*: cada conexão é definida por um *Input Logical Link* (ILL) ou seja, uma fonte de dados, e um *Output Logical Link* (OLL) que representa um destino para os dados pós processamento. A exceção de quando o ILL é o IP e quando o OLL é um OP, um nome para a ponte e os MACs das interfaces que receberão se conectarão a mesma devem ser definidos.

A validação de grafo utiliza regras para garantir que existe um grafo totalmente conectado e sem possíveis *loopings* infinitos, que o IP contenha apenas uma conexão com uma VNF afim de enviar dados (OLL) e nunca sejam utilizados como entrada de dados (ILL), que os OPs sejam utilizados como entrada de dados por VNFs (ILL) e nunca sejam utilizados como saída de dados (OLL), além de garantir que todos os elementos previamente definidos figurem no grafo da SFC.

São dispostos métodos para controle da SFC como um todo com ações padrão como ligar, desligar, criar, modificar e destruir, além de possibilitar que o gerente da cadeia de funções de rede virtualizadas acesse individualmente cada VNF afim de utilizar sua respectiva interface REST através da API e permitir a submissão de *scripts* de execução a serem aplicados a SFC como um todo.

3.4. *Scripts* de Execução

Um *script* de execução representa ações de alto nível descritas através de um conjunto de chamadas REST realizadas de forma serial destinadas a uma VNF específica.A execução

do *script* é realizada atravésda classe SFC com a identidicação da ação a ser excutada. A instância da SFC pesquisa todas as VNFs presentes no *script* que estão ativas e que contenham a ação requisitada e repassa os comandos a serem executados. A definição de uma ação de exceção em caso de erro é opcional.

A definição de um *script* de execução é dada por:

- *ID*: como um *script* de execução pode determinar ações diferentes para VNFs diferentes, o ID da VNF deve ser explícito determinando a quem se destina as definições;
- *Function*: função de rede virtualizada escrita em Click que será levada em consideração para a realização das ações;
- *Path*: local onde o arquivo da função nomeada no item anterior está localizada;
- *Actions*: consiste de um identificador da ação que pode ser requisitada por um agente externo, cada ação consiste de um conjunto de chamadas REST adicionada de sua precedência.

Como alternativa, um dicionário pode ser passado a classe VNF contendo apenas as chamadas REST e sua precedência para serem executadas de maneira serial, nesse caso, dispensa-se a definição de um arquivo JSON para o script e realiza-se as requisições puramente via código.

4. Conclusão

Com a crescente popularização de NFV, também cresce a demanda por *enablers* capazes de manter e executar uma infraestrutura baseada nesta tecnologia. Click-On-OSv é uma plataforma capaz de gerenciar e executar funções de rede virtualizadas implementadas em Click, sendo que eu diferencial reside na disponibilização de interfaces Web e REST nativa de gerencia.

Este trabalho apresenta uma API em Python que objetiva facilitar o gerenciamento tanto de uma VNF específica quanto da SFC como um todo. A API conta com quatro módulos que se complementam, são eles o módulo de VM, de VNF, de SFC e de *Scripts* de Execução.

Em trabalhos futuros objetiva-se aprimorar a API para trabalhar com VNFs compartilhadas, permitindo a utilização da mesma VNF entre duas ou mais SFCs. A flexibilização das tecnologias para instanciação de VMs e conexão entre as mesmas, permitindo a utilização de diferentes *hypervisors* e de Open vSwitch, também é finalidade deste projeto.

Ainda como trabalhos futuros, alterações no módulo de execução de *scripts* serão projetados a fim de permitir a definição de uma exceção global, dessa forma ações como de *rollback* do sistema podem ser consideradas. Por fim, objetiva-se flexibilizar a definição das sequências de ações através da hierarquização de processos considerando um cenário de sucesso ou falha para cada instrução.

Referências

Baton, O. (2017). Open baton: An open source reference implementation of the etsi mano. `https://openbaton.github.io`. Accessed: 2017-07-11.

Csoma, A., Sonkoly, B., Csikor, L., Németh, F., Gulyas, A., Tavernier, W., and Sahhaf, S. (2014). Escape: Extensible service chain prototyping environment using mininet, click, netconf and pox. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 125–126. ACM.

ETSI, G. (2013). Network functions virtualisation (nfv); use cases. *V1*, 1:2013–10.

ETSI, N. (2012). Network functions virtualization, white paper.

Hwang, J., Ramakrishnan, K. K., and Wood, T. (2015). Netvm: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management*, 12(1):34–47.

Intel (2017). Intel data plane development kit. `http://http://dpdk.org`. Accessed: 2017-05-27.

Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297.

Marcuzzo, L. d. C., Garcia, V. F., Cunha, V., Corujo, D., Barraca, J. P., Aguiar, R. L., Schaeffer-Filho, A. E., Granville, L. Z., and Santos, C. R. P. d. (2017). Click-on-osv: A platform for running click-based middleboxes. In *2017 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE.

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 *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 459–473, Seattle, WA. USENIX Association.

Mechtri, M., Ghribi, C., Soualah, O., and Zeghlache, D. (2017). Nfv orchestration framework addressing sfc challenges. *IEEE Communications Magazine*, 55(6):16–23.

OSv (2017). Osv - the operating system. `http://osv.io`. Accessed: 2017-07-11.

Quinn, P. and Nadeau, T. (2015). Problem statement for service function chaining.

Ramakrishnan, K. (2016). Software-based networks: Leveraging high-performance nfv platforms to meet future communication challenges. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 24–24. IEEE.

Rizzo, L. (2012). Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112.

Sefraoui, O., Aissaoui, M., and Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3).

Tacker (2017). Tacker: Openstack nfv orchestration. `https://wiki.openstack.org/wiki/Tacker`. Accessed: 2017-07-12.

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.