Estudo Comparativo de Controladores Software-Defined Networking (SDN)

Comparative Study of Software-Defined Networking (SDN) Traffic Controllers

Gonçalo Pereira, José Silva
Department of Informatics
University of Minho
Braga, Portugal
a74413@alunos.uminho.pt, a75280@alunos.uminho.pt

Pedro Sousa
Centro Algoritmi, Department of Informatics
University of Minho
Braga, Portugal
pns@di.uminho.pt

Resumo — **As redes SDN (*Software-defined Networking*) vieram introduzir um novo paradigma de controlo nas infraestruturas das redes de comunicação, tendo pois um grande impacto não só na operação da rede em si, mas também em todo o desenvolvimento realizado em torno das mesmas (e.g. serviços de rede, serviços distribuídos, aplicações Internet, etc.). Assim, e por forma a auxiliar os investigadores e profissionais destas áreas, este artigo apresenta um estudo comparativo entre vários controladores SDN existentes no mercado. Numa primeira fase é feita uma análise mais alargada sobre diversos controladores, destacando-se as suas caraterísticas gerais e salientando as áreas onde foram desenvolvidos alguns trabalhos científicos usando cada um dos controladores. Numa segunda fase, é feita uma seleção de alguns controladores, sendo implementados alguns microprojectos ilustrativos, o que permitiu efetuar um estudo comparativo no que que se refere à curva de aprendizagem e algumas das principais *APIs* suportadas pelos controladores.**

Palavras Chave – Redes de Computadores, SDNs, OpenFlow;

Abstract — **The SDN (Software-defined Nettworking) has introduced a new control paradigm in network infrastructures, thus having a great impact not only on the operation of the network itself, but also on supporting all the development carried out around them (e.g. network and distributed services, Internet applications, etc.). In this context, and in order to assist researchers and professionals in these areas, this article presents a comparative study among several SDN controllers in the market. In a first phase, a more general analysis is made on several controllers, highlighting their general characteristics and the areas where some scientific works were developed using each of the controllers. After that, a selection of some controllers is made, and some illustrative microprojects were implemented, allowing to obtain a comparative analysis of the learning curves and some of the main APIs supported by each SDN controller.**

Keywords – Computer Networks, SDNs, OpenFlow;

I. INTRODUÇÃO

As estruturas das redes de comunicação atuais são bastante complexas pelo que a sua manutenção e desenvolvimento de novas funcionalidades e serviços não são triviais. As redes *Software-Defined Networking* (SDN) [1,2] separaram claramente os planos de controlo e de dados, e centralizam a visão global da rede num único componente, o controlador SDN. Esta abordagem inovadora facilita diversas tarefas de gestão, bem como potencia e agiliza o desenvolvimento e a programação de novos serviços e aplicações Internet mais versáteis sobre as infraestruturas de comunicação.

No entanto, a transição para o mundo das SDNs pode não ser um processo fácil para os profissionais e investigadores desta área. A elevada diversidade dos controladores SDN existentes, bem como as suas heterogéneas caraterísticas e potencialidades, são fatores que dificultam a escolha das plataformas que melhor se adequam a um determinado objectivo. Este trabalho posiciona-se exatamente neste contexto, tentando dar resposta e alavancar este primeiro passo, ou seja, a escolha do controlador SDN a ser utilizado num determinado projeto. Desta forma, este trabalho de investigação pretende auxiliar os investigadores nesta área, fornecendo-lhes inicialmente uma caraterização geral de alguns controladores SDN e as áreas temáticas onde eles foram utilizados. Posteriormente, para um conjunto restrito de controladores é indicado o tipo de curva de aprendizagem associada aos controladores e descritas algumas das APIs por eles suportadas.

De seguida, na Secção II, é feita uma contextualização ao funcionamento geral das SDN, sendo posteriormente descritos na Secção III vários controladores SDN existentes no mercado. Na Secção IV é apresentada uma análise comparativa entre os vários controladores discutidos, analisando diferentes parâmetros dos mesmos. A Secção V apresenta pequenos micro-projectos desenvolvidos num conjunto mais limitado de controladores, a partir dos quais é apresentado um estudo comparativo da curva de aprendizagem e *APIs* suportadas pelos mesmos. Por fim a Secção VI apresenta as conclusões.

II. SOFTWARE-DEFINED NETWORKING

As SDNs separam a lógica de controlo da rede (plano de controlo) das camadas inferiores associadas aos *routers* e *switches* responsáveis pelo encaminhamento do tráfego (plano

de dados). Desta forma, a lógica de controlo fica centralizada num único elemento, o controlador. Podemos pois encarar as SDNs como sendo uma arquitetura onde temos o controlador como ponto centralizado da inteligência da rede, sendo ele que garante a comunicação entre os *switches/routers* e as aplicações ou lógicas que controlam a operação da rede. Para comunicar com a camada superior, o controlador usa as denominadas *Northbound Interfaces* (NBIs). Estas interfaces vão permitir a um programador programar a rede e controlá-la tendo em vista um determinado objectivo associado à aplicação em causa. Outro componente bastante importante nas SDNs são as *Southbound Interfaces* (SBIs), sendo estas responsáveis por interagir com a camada inferior. Esta interface permite que o controlador defina o comportamento dos equipamentos de rede. Atualmente, a interface mais difundia utiliza o protocolo *OpenFlow*. No entanto, neste contexto, existem muitos outros, como o NETCONF, RESTCONF, *Extensible Messaging and Presence Protocol* (XMPP) ou o *Open vSwitch Database Management Protocol* (OVSDB).

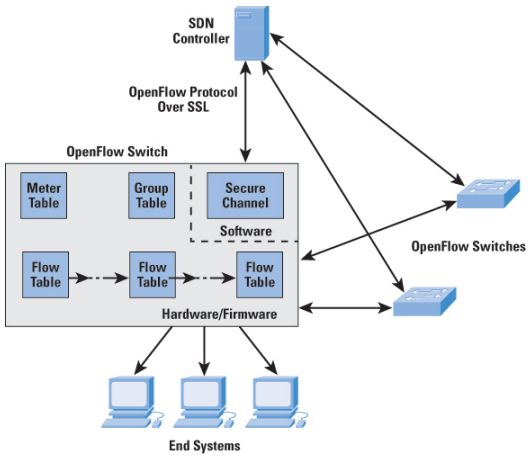

Figura 1. Componentes de um Switch OpenFlow [29]

O *OpenFlow* foi criado pela *OpenFlow Network Foundation* (ONF) e foi um dos primeiros protocolos desenhado para ser adotado pelas SDNs. Em termos gerais, este protocolo permite aos controladores determinar e verificar o caminho dos pacotes de rede até ao destino numa rede constituída por *switches*. Além disso, permite que múltiplos equipamentos de rede de marcas diferentes possam ser geridos apenas por um único protocolo aberto. Num ambiente *OpenFlow*, qualquer equipamento que queira comunicar com o controlador necessita de suportar este protocolo. Uma vez verificado, é feita uma ligação entre controlador e *switch/router* por *Secure Sockets Layer* (SSL), tendo o controlador de fazer fluir o tráfego de rede, podendo para tal adicionar, remover e atualizar as entradas de *switching* ou encaminhamento associadas a um ou mais fluxos de dados da rede.

Como se apresenta na Figura 1, um *switch OpenFlow* possui três tabelas: tabela de fluxo *(flow table)*, tabela de grupo *(group table)* e tabela de medição *(meter table)*. O plano de dados de um comutador *openflow* é um pipeline com uma ou mais *tabelas de fluxo*, cujas entradas são configuradas pelo controlador. Cada entrada de fluxo consiste de campos que permitem identificar um fluxo e um conjunto de instruções a aplicar aos pacotes desse fluxo. Se um pacote obedece aos campos de uma entrada, as ações associadas à entrada serão aplicadas ao pacote. As ações podem enviar o pacote para alguma das tabelas seguintes, ou descrever processos de encaminhamento do pacote (e.g. enviar o pacote para uma porta de saída), ou ainda especificar processamento por grupos ou mecanismos de medição. O processamento por grupo ou processo de medição direciona o pacote para a tabela de grupo ou tabela de medição, respectivamente. Sumariamente, grupos representam conjunto de ações usadas em processos de encaminhamento mais complexos, enquanto as entradas de medição permitem, por exemplo, que o *switch* controle a taxa de bits dos fluxos que o atravessam.

III. Controladores SDN

Atualmente existem diversos controladores SDN [3] que utilizam o *OpenFlow* ou outros protocolos no seu funcionamento. Por motivos de limitação do tamanho deste artigo, nesta secção será apresentada unicamente uma muito pequena descrição inicial de alguns dos controladores existentes no mercado. Posteriormente, na Secção IV, alguns destes controladores serão alvo de uma caraterização e comparação mais exaustiva em diferentes parâmetros.

A. *Floodlight*

Este controlador nasceu num ambiente mais dedicado à área empresarial e por isso é um dos mais usados neste contexto. Além disso uma das principais razões da sua utilização é porque explora a questão da escalabilidade, sendo apontado como um dos mais escaláveis no mercado. Desenvolvido em Java, possui suporte multiplataforma e possui licença *open-source* fornecida pela Apache.

B. *Beacon*

Este controlador foi proposto pela Universidade de *Standford* em 2010 e tem como base o *Floodlight*. Como é baseado em Java suporta multiplataforma e conta ainda com operações baseadas em *multithreading* e em eventos.

C. *IRIS*

Este controlador é baseado nos controladores *Floodlight* e *Beacon*. Este pretende não só melhorar esses dois controladores como propõe horizontalidade nas redes, alta disponibilidade e transparência caso ocorra alguma falha, bem como suporte multi-domínio abstrato baseado no *OpenFlow*.

D. *FlowER*

Este controlador é baseado na linguagem *Erlang*, uma linguagem de programação funcional, e foi criado para oferecer uma plataforma especialmente focada na lógica do *switch* e no fluxo de dados.

E. *Loom*

Implementa os protocolos OpenFlow 1.3.x e 1.4 com o intuito de explorar a escalabilidade e robustez dos controladores. Está implementado em *Erlang* para permitir uma fácil compreensão e desenvolvimento de aplicações

F. *Maestro*

Ao contrário dos controladores anteriormente mencionados, trata-se de um sistema operativo e não apenas

de um controlador. Esta solução está bastante focada na questão do paralelismo no controlo de rede para melhorar o desempenho do sistema. Além disso, este controlador permite utilizar outro tipo de protocolos que não o *OpenFlow*.

G. NOX

O NOX foi um dos primeiros controladores a implementar o protocolo OpenFlow 1.0. Este controlador permite o desenvolvimento na linguagem C++, possibilitando um ambiente assíncrono e baseado em eventos. Tem sido um controlador muito utilizado em ambiente académico para desenvolvimento de aplicações, contudo a sua atividade tem apresentado um decréscimo devido ao seu sucessor, o POX.

H. POX

Trata-se de uma variante do NOX que começou a ser mais utilizada por suportar uma vista da topologia em grafos e fornecer suporte para virtualização. A sua linguagem de programação é o *Python*, oferecendo igualmente uma plataforma de desenvolvimento bastante rápida.

I. OpenDayLight (ODL)

Este controlador foi criado para divulgar e promover as SDNs e as funções virtuais de rede (*Networks Functions Virtualization*). Sobre a alçada da *Linux Foundation*, este controlador permite não só trabalhar em ambiente empresarial, como em ambiente académico permitindo bastantes propósitos. Além de ter *OpenFlow* implementado, permite outro tipo de protocolos tornando-o um dos mais utilizados neste sentido. Foi desenvolvido em *Java* por forma a abranger múltiplas plataformas seguindo a sua filosofia de abertura para todos os utilizadores.

J. Ryu

Tal como o *OpenDayLight*, o *Ryu* também permite multiprotocolo com a diferença da sua linguagem ser *Python*. Este controlador permite controlo de eventos, controlo de aplicações, serviço de mensagens e oferece uma série de livrarias reutilizáveis para outros protocolos.

K. Faucet

Este controlador é baseado em *Python* e tem como base o *Ryu*. Possui duas componentes principais: *Faucet* e *Gauge*. A primeira é responsável por todo o controlo de rede e a segunda é mais orientada a processos de monitorização da infraestrutura.

L. NodeFlow

Desenvolvido pela Cisco, este controlador está implementado em *JavaScript* para *Node.JS* permitindo uma solução rápida no contexto SDN. Um dos aspetos mais reforçados na sua apresentação, é que por ser desenvolvido em *Node.JS* não necessita de muitos recursos computacionais e daí resultar uma solução leve e escalável.

M. Open Networking Operating System

Este sistema operativo tem como objetivo oferecer escalabilidade, alta disponibilidade e alto desempenho aos fornecedores de serviços de comunicações. Apesar de ser específico para essas situações, pode também servir como um controlador SDN. Este controlador permite um controlo em nuvem pelo que consegue trazer novas aplicações mais rapidamente porque não precisa de alterar o plano de dados. Além disso a sua configuração e controlo em tempo real faz com que não sejam necessários protocolos de *routing* e/ou de *switching*. Outro pormenor que o torna bastante usado é o facto de pertencer à *Linux Foundation* que possui meios de divulgação bastante desenvolvidos.

N. OpenMul

Solução que permite controlar equipamentos que utilizem como protocolo o *OpenFlow*, o OVSDB e o NETCONF. Desenvolvido de raiz em C, é bastante escalável quando comparado com outros controladores e carateriza-se pelo seu elevado desempenho, através de baixas latências e elevadas velocidades de transferências de dados entre o controlador e os equipamentos da rede.

O. Trema

O Trema é uma *framework* que permite criar controladores que usam OpenFlow nas linguagens C e *Ruby*. Apresenta-se como tendo sido desenvolvido a pensar na simplicidade de código conciso por forma a facilitar a sua manutenção e assim reduzir a ocorrência de possíveis *bugs*.

P. Rosemary

Este controlador possui uma arquitetura baseada em *containers* denominada de micro-NOS (*NOS: Network Operating System*). Carateriza-se por oferecer um isolamento e segurança às aplicações para que no caso de falha não se propagarem pela *stack* da SDN.

IV. Análise Comparativa

Nesta secção será apresentado um estudo comparativo entre alguns dos controladores anteriormente discutidos. Para tal, foram analisadas diversas fontes de informação, o que também incluiu uma análise cuidada a muita da documentação de apoio e suporte associada aos diversos controladores estudados. Como resultado desta análise comparativa, a Tabela I apresenta e compara diversos aspetos caraterizadores dos vários controladores que são deveras relevantes para a sua utilização.

Na Tabela I são analisadas diversas caraterísticas dos controladores, tais como: ano de lançamento, a linguagem de programação suportada, plataformas para as quais se encontram disponíveis, tipo de interface, classificação da sua documentação sendo atribuídos três níveis: ***Fraco***, quando apenas é disponibilizada documentação de âmbito geral sobre o controlador, ***Razoável***, quando além da documentação geral, existe uma descrição das funções das APIs disponibilizadas (e.g. funções, classes, métodos, etc.) e ***Bom***, quando, além dos pontos anteriores, se apresentam também tutoriais de iniciação e demonstrações e exemplos concretos de utilização das APIs suportadas pelo controlador. Outro aspeto analisado na Tabela I é a versão do *OpenFlow* suportada pelo controlador.

Adicionalmente, na Tabela I, e a título ilustrativo, referem-se também alguns trabalhos desenvolvidos usando os controladores em determinadas áreas de investigação no campo

das redes de computadores. Neste contexto, a quantidade de referencias apresentadas acabam também por refletir um pouco o nível e a heterogeneidade da utilização de cada um dos controladores. Acreditamos que está categorização adicional será bastante útil para os investigadores/programadores SDN que pretendam desenvolver trabalhos nas áreas indicadas, podendo pois utilizar algumas das referencias apresentadas como ponto de partida para o desenvolvimento de trabalhos em cada uma das áreas abrangidas. Neste contexto, as áreas selecionadas neste estudo, e que são referidas na Tabela I são: ***ET****: Engenharia de Tráfego*, ***M****: Monitorização de Redes*, ***RDC****: Redes de Data Centers*, ***S****: Segurança de Redes* e ***W****: Gestão de Redes Wireless*.

V. MICRO-PROJETOS ILUSTRATIVOS

Nesta secção serão apresentados sumariamente alguns microprojectos que foram desenvolvidos utilizando uma seleção mais restrita de quatro controladores SDN (*Floodlight, POX, ONOS e ODL*). O foco aqui, não foi tanto a temática em si de cada um dos micro-projetos selecionados, mas antes todo o processo e etapas que foi necessário ultrapassar para se obter uma implementação completa de um determinado mecanismo na rede SDN. Estes micro-projetos atravessaram todas as fases com que se enfrentará qualquer investigador/programador no mundo das SDNs, nomeadamente: *i)* familiarização inicial com o controlador, *ii)* instalação do controlador, *iii)* leitura de documentação auxiliar, *iv)* exploração das APIs disponibilizadas, *v)* codificação/programação do mecanismo a implementar e *vi)* testes a análise de resultados. Mediante todo este processo foi possível adquirir o conhecimento relativo a dois parâmetros importantes relativamente a cada um dos controladores: *i)* o tipo de curva de aprendizagem subjacente ao desenvolvimento de projetos usando o controlador em questão e *ii)* uma descrição geral de algumas das funcionalidades para os quais o controlador disponibiliza APIs específicas.

Para a execução dos projetos foi utilizado o emulador de rede *Mininet* [30]. Esta plataforma emula redes de *switches* SDNs que podem comunicar com um ou mais controladores SDN locais ou remotos. Neste caso, por questões de desempenho, optou-se por trabalhar com uma máquina virtual fornecida pelo *Mininet* que se interliga com uma outra máquina virtual que contém todos os controladores utilizados nos micro-projectos. Ambas as máquinas virtuais encontram-se ligadas por uma rede do tipo *host-only*. Quanto aos controladores utilizados, estes encontram-se instalados em máquinas virtuais com sistemas operativos baseados em UNIX (*Ubuntu* e *Xubuntu*), sendo disponibilizadas pelos programadores dos controladores (*FloodLight* e *ONOS*) ou por comunidades de investigação nesta área (*POX* e *ODL*).

A. Floodlight – Mecanismo de Controlo de Acessos

Neste primeiro micro-projeto foi implementado um módulo no controlador *Floodlight* que permite definir um conjunto de regras que, de uma forma (re)configurável, bloqueiam o acesso entre determinados *hosts* da infraestrutura de rede. Isto permite, por exemplo, isolar determinados *hosts* de outras máquinas da rede, bem como condicionar as comunicações entre determinados conjuntos de máquinas, quando tal se torna necessário (ver a Figura 2 a)).

Como referido, o controlador *Floodlight* encontra-se desenvolvido em Java e está dividido em módulos sendo que estes serão carregados aquando a ligação do controlador à topologia *Mininet*. Também se destaca a versatilidade da programação no controlador *Floodlight*, visto que, ao ser criado um novo módulo, este contém já todos os seus tipos primitivos declarados e prontos a serem usados, permitindo a fácil manipulação de dados do mesmo. Após a finalização do projeto foi possível verificar a correta operação do módulo desenvolvido, sendo possível estabelecer diferentes regras de bloqueamento de comunicações entre os *hosts* e comprovar a eficácia das mesmas através do emulador de rede *mininet*.

B. POX – Balanceamento de Carga

Como segundo micro-projeto, o controlador POX foi utilizado para a implementação de um módulo que permitisse realizar balanceamento de carga entre vários servidores HTTP (ver a Figura 2 b)). As técnicas de balanceamento de carga dão a possibilidade de responder com eficácia a pedidos de clientes sem que os servidores fiquem sobrecarregados. O balanceamento parte de regras definidas no controlador para que o tráfego seja distribuído por todos os servidores na rede e obter melhores tempos de resposta e utilização destes. Apesar de anteriormente ter-se constatado que a documentação deste controlador é fraca, a sua aprendizagem é facilitada por possuir diversos módulos pré-carregados em *Python* com algumas funcionalidades base, bastando ajustá-las e adaptar para aquilo que se pretende. Neste caso, num módulo já existente, o processo de balanceamento de carga era realizado de forma aleatória, sendo no entanto bastante limitado numa

TABELA I. ANÁLISE COMPARATIVA DOS CONTROLADORES

Controlador	**Parâmetros Comparativos**						
	Ano	***Linguagem de Programação***	***Plataforma***	***Interface***	***Documentação***	***Versão OF***	***Exemplos de Áreas de Trabalhos Desenvolvidos***
Beacon	2010	Java	Linux, MAC OS e Windows	Web	Razoável	OpenFlow 1.0	ET [4], M [5], W [6] e S [7]
IRIS	2014	Java	Linux, MAC OS e Windows	Baseado em Web e Java	Razoável	OpenFlow 1.0.1 até 1.3.2	RDC [8]
NOX	2008	C++	Linux	PyQT4	Fraco	OpenFlow 1.0	ET [9], M [10], RDC [11] e S [12]
POX	2012	Python	Linux, MAC OS e Windows	PyQT4	Fraco	OpenFlow 1.0, parcialmente 1.1	ET [13], M [14] e S [15]
Floodlight	2012	Java	Linux, MAC OS e Windows	Baseado em Web e Java	Bom	OpenFlow 1.0 até 1.5	ET [16], W [17], M [18], RDC [19] e S [20]
OpenDayLight	2013	Java	Linux, MAC OS e Windows	Baseado em Web	Bom	OpenFlow 1.0 até 1.4	ET [21], M [22], RDC [23]
ONOS	2015	Java	Linux, MAC OS e Windows	Baseado em Web	Bom	OpenFlow 1.0 até 1.5	M [24]
Ryu	2013	Python	Linux	GUI Patch	Razoável	OpenFlow 1.0 até 1.5	ET [25]
OpenMUL	2012	C	Linux	Baseado em Web	Razoável	OpenFlow 1.4	RDC [26]
Trema	2011	C/C++ e Ruby	Linux	N/A	Fraco	OpenFlow 1.0 até 1.3	ET [27]

perspectiva de utilização real, pelo que se decidiu enriquecer a semântica do balanceamento de carga efetuado.

Consequentemente, este módulo foi modificado para que seguir regras pré-definidas pelo administrador. No exemplo implementado optou-se por uma regra horária, ou seja, durante um determinado intervalo temporal o controlador encaminha

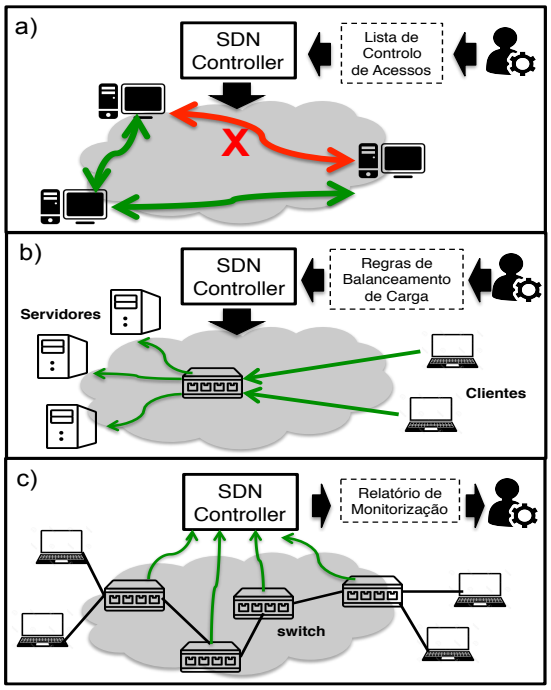

Figura 2 Micro-projetos desenvolvidos via controladores SDN

todo o tráfego para um determinado servidor pré-definido. Finalizado esse período é verificada a existência de nova(s) regra(s) de balanceamento, e seleciona-se o servidor que prestará serviço durante o novo intervalo temporal em causa.

No final deste micro-projeto foi possível verificar a eficácia do módulo desenvolvido no emulador através de uma rede que envolvia vários servidores e clientes a realizar pedidos HTTP, sendo estes atendidos por diferentes servidores em diferentes intervalos temporais, Figura 2 b).

C. ONOS e ODL – Monitorização de Redes

Por fim desenvolveu-se um micro-projecto envolvendo os controladores ONOS e ODL, ambos da *Linux Foundation,* tendo bastante adesão no desenvolvimento de aplicações. Para ambos, o objetivo principal seria compreender e explorar o desenvolvimento de aplicações visto serem bastante divulgados e possuírem documentação bem estruturada. Foi decidido desenvolver um módulo de monitorização de rede (ver a Figura 2 c)), aproveitando o facto do controlador possuir uma visão geral sobre a topologia da rede, fazendo assim sentido obter estatísticas desta e assim poder geri-la da melhor forma. Esta aplicação de monitorização não é suportada pelo controlador de raiz, i.e. os seus módulos base não fornecem uma funcionalidade integral deste género, pelo que a aplicação teve que ser construída através da manipulação de várias funções das APIs disponibilizadas.

No fim da implementação deste micro-projeto foi possível apresentar diversas métricas relativas às portas de saída dos *switches* intervenientes na topologia do emulador *mininet*. Estas estatísticas vão desde débitos dos fluxo, número de pacotes recebidos e enviados pelas interfaces dos *switches*, número de pacotes perdidos, pacotes recebidos com erro, número total de pacotes e de bytes enviados, entre outras métricas. Estas estatísticas eram armazenadas nos *logs* gerados por ambos os controladores, sendo posteriormente apresentadas ao administrador de rede através de um interface, Figura 2 c).

D. Curva de Aprendizagem e APIs Suportadas

Como referido, a pesquisa e desenvolvimento subjacente aos projetos descritos permitiu a obtenção de informação relevante relativamente ao tipo de curva de aprendizagem inerente aos controladores, bem como informação geral de algumas das funcionalidades para os quais os controladores disponibilizam APIs específicas. Na Tabela II é sumariada toda esta informação para os controladores SDN considerados.

VI. Conclusões

O aparecimento das SDNs irá alterar consideravelmente todo o processo de gestão das infraestruturas de comunicação, bem como das aplicações e serviços que sobre elas se podem desenvolver. Este trabalho pretende contribuir para uma mais rápida adaptação e conhecimento dos investigadores e profissionais da área das redes a este novo paradigma.

Neste contexto, o trabalho apresentado centrou-se em dois planos. Inicialmente, num plano mais teórico, onde através de investigação de várias fontes de informação foi apresentada um categorização mais geral de vários controladores SDNs existentes no mercado. Esse estudo incluiu ainda a identificação de várias áreas temáticas onde foram desenvolvidos projetos envolvendo os controladores estudados, tendo sido apresentados vários trabalhos que podem constituir um ponto de partida útil para os interessados nessas áreas de investigação. O trabalho incluiu também um plano mais experimental. Neste, e para um conjunto mais restrito de controladores, foram desenvolvidos vários micro-projetos ilustrativos que possibilitaram o contacto dos investigadores com as diferentes plataformas e a programação usando esses controladores. Dessa experiência mais prática, obteve-se informação bastante relevante sobre as curvas de aprendizagem dos controladores e uma descrição de algumas funcionalidades suportadas pelas APIs disponibilizadas.

Agradecimentos

Este trabalho foi apoiado pela FCT - Fundação para a Ciência e Tecnologia no âmbito do Projeto UID/CEC/00319/2019.

Referências Bibliográficas

[1] T. D. Nadeau and K. Gray., “SDN: Software Defined Networks”, O’Reilly, 2013.

[2] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, “Software-defined networking: A 1comprehensive survey”. Proceedings of the IEEE, 103(1):14–76, 2015.

TABELA II. CURVA DE APREENDIZAGEM E APIs SUPORTADAS PELOS CONTROLADORES SDN

Controlador	**Parâmetros Avaliados**	
	Curva de Aprendizagem (Rápida, Moderada, Lenta)	***Exemplos de APIs Suportadas***
Floodlight	Aprendizagem facilitada fruto da sua boa documentação; Alta versatilidade pela simplicidade de adição e criação de módulos; Boa interface grãfica que apresenta a topologia e alguns módulos capazes de detalhar certas características. [Curva de Aprendizagem: *Rápida*]	Monitorização, Obtenção de Estados do Controlador e dos *switches*, Estatísticas de *OpenFlow*, Estatísticas de Rede, Descoberta de Topologia e de Encaminhamento, Gestão de Fluxo, Criação de Redes Virtuais, Gestão de *Firewall*, Controlo de Acesso, Monitorização de *Performance*, Balanceamento de Carga.
POX	Documentação relativamente limitada, porém bem estruturada; integração de módulos simplificada, bastando apenas especificar a *path* para o módulo. [Curva de Aprendizagem: *Moderada*]	Descoberta de Topologia, Gestor de Fluxo, Balanceamento de Carga, *Messenger*, Aprendizagem de rotas imitando um *switch* de nível 2 ou 3, Definição de Regras de Controlo de Acesso.
ONOS	Boa documentação, capaz de tornar a experiência do utilizador mais intuitiva no que toca à resolução de problemas; instalação de módulos facilitada pela utilização dos comandos que possui. [Curva de Aprendizagem: *Moderada*]	Gestão de Dispositivos, Gestão de Ligações, Gestão de *hosts,* Descoberta de Topologia, Descoberta de Caminhos, Gestão de Fluxos, Diretiva de Fluxo Objetivo que permite abstração entre fluxo e pipeline dos dispositivos na tabela, Gestão de Grupos, Gestão de Métricas, Gestão de Aplicações, Configuração de Componentes.
OpenDayLight	Boa documentação mas parcialmente desatualizada aquando das atualizações para novas versões; criação de módulos exige conhecimento profundo em *Maven*. [Curva de Aprendizagem: *Lenta*]	Descoberta de Topologia, Rastreamento de *Hosts*, Gestão de Fluxo, Estatísticas, *Subnetting*, Routing Estático, Gestor de *Switches*, Gestor de Utilizadores, Gestor de *Containers*, Gestor de Conexões, Construção de modelos de Bridge Domain via yang/netconf, Suporte do Openstack por Neutron ML2.

[3] O. Salman, I.H. Elhajj, K. Ayman, and A. Chehab, "Sdn controllers: A comparative study", 2016 18th Mediterranean Electrotechnical Conference (ME-LECON), 2016.

[4] P. Xiong, H. Hacigumus and J. F. Naughton. "A software-defined networkingbased approach for performance management of analytical queries on distri-buted data stores". In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pages 955–966, New York,NY, USA, 2014. ACM.

[5] D. B. Hoang and M. Pham,"On software-defined networking and the designof sdn controllers". In 2015 6th International Conference on the Network of the Future (NOF), pages 1–3, Sept 2015.

[6] S. A. Shah et al. "An architectural evaluation of sdn controllers." In 2013 IEEE International Conference on Communications (ICC), pages 3504–3508, June 2013.

[7] A. Lara, A. Kolasani, and B. Ramamurthy. "Network innovation using openflow: A survey". IEEE Communications Surveys Tutorials, 16(1):493–512, First 2014.

[8] J. Shin, T. Kim, B. Lee and S. Yang. "Iris-hisa: Highly scalable and avai-lable carrier-grade sdn controller cluster". Mobile Networks and Applications, 22(5):894–905, Oct 2017.

[9] Nikhil Handigol, Srini Seetharaman, Mario Flajslik, Aaron Gember, NickMcKeown, Guru Parulkar, Aditya Akella, Nick Feamster, Russ Clark, ArvindKrishnamurthy and others. "Aster* x: Load-balancing web traffic over wide-area networks", 11 2010.

[10] A. Tootoonchian, M. Ghobadi and Y. Ganjali. "Opentm: Traffic matrix estimator for openflow networks". In Proceedings of the 11th International Conference on Passive and Active Measurement, PAM'10, pages 201–210, Berlin,Heidelberg, 2010. Springer-Verlag.

[11] T. Benson, A. Akella, A. Shaikh and S. Sahu. "Cloudnaas: A cloud networking platform for enterprise applications". In Proceedings of the 2Nd ACMSymposium on Cloud Computing, SOCC '11, pages 8:1–8:13, New York, NY,USA, 2011. ACM.

[12] R. Braga, E. Mota, and A. Passito. "Lightweight ddos flooding attack detectionusing nox/openflow". In Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks, LCN '10, pages 408–415, Washington, DC, USA, 2010. IEEE Computer Society.

[13] M. S. Seddiki et al. "Flowqos: Qos for the rest of us". In Proceedings of the ThirdWorkshop on Hot Topics in Software Defined Networking, HotSDN '14, pages207–208, New York, NY, USA, 2014. ACM.

[14] N. L. M. van Adrichem, C. Doerr and F. A. Kuipers. "Opennetmon: Networkmonitoring in openflow software-defined networks." In 2014 IEEE Network Operations and Management Symposium (NOMS), pages 1–8, May 2014.

[15] S. Shin, V. Yegneswaran, P. Porras and G. Gu. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks". In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, pages 413–424, New York, NY, USA, 2013. ACM.

[16] H. E. Egilmez, S. T. Dane, K. T. Bagci and A. M. Tekalp. "Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks". In Proceedings of The 2012 AsiaPacific Signal and Information, pages 1–8, Dec 2012.

[17] J. Schulz-Zander, N. Sarrar, and S. Schmid. "AeroFlux: A near-sighted controller architecture forsoftware-defined wireless networks". In Presented aspart of the Open Networking Summit 2014 (ONS 2014), Santa Clara, CA,2014. USENIX.

[18] J. Suh, T. T. Kwon, C. Dixon, W. Felter and J. Carter. "Opensample: A low-latency, sampling-based measurement platform for commodity sdn". In 2014 IEEE 34th International Conference on Distributed Computing Systems,pages 228–237, June 2014.

[19] E. Keller, S. Ghorbani, M. Caesar and J. Rexford. "Live migration of an entire network (and its hosts)". In Proceedings of the 11th ACM Workshop on HotTopics in Networks, HotNets-XI, pages 109–114, New York, NY, USA, 2012.ACM.

[20] R. Hand, M. Ton and E. Keller. "Active security". InTwelfth ACM Workshopon Hot Topics in Networks (HotNets-XII), 2013.

[21] T. Kim et al. "Load balancing on distributed datastore in opendaylight sdn controller cluster". In2017 IEEE Conference on Network Softwarization (NetSoft), pages 1–3, July 2017.

[22] Z. K. Khattak, et al. "Performance evaluation of opendaylight sdn controller". In 2014 20th IEEE International Conference on Paralleland Distributed Systems (ICPADS), pages 671–676, Dec 2014.

[23] D. Suh, S. Jang, S. Han, S. Pack, T. Kim and J. Kwak. "On performance of opendaylight clustering". In 2016 IEEE NetSoft Conference and Workshops(NetSoft), pages 407–410, June 2016.

[24] W. Kim, J. Li, J. W. K. Hong and Y. J. Suh. "Ofmon: Openflow monitoring system in onos controllers". In 2016 IEEE NetSoft Conference and Workshops(NetSoft), pages 397–402, June 2016.

[25] H. Kashi and H. Ajorloo. "A framework for application-aware networking by delegating traffic management of sdns".CoRR, abs/1610.05062, 2016.

[26] Chinmay Mhatre. "Scalability Testing of SDN Controllers and Switches". PhDthesis, Department of Computer Science and Engineering Indian Institute of Technology, Bombay, 2015.

[27] K. Hiroya S. Kazuya. "An openflow controller for reducing operational cost of ip-vpns".NEC Technical Journal/Vol.8 No.2/Special Issue on SDN and Its Impact on Advanced ICT Systems, 2014.

[28] B. Lee, S. H. Park, J. Shin, and S. Yang. "Iris: The openflow-based recursive sdn controller". In16th International Conference on Advanced CommunicationTechnology, pages 1227–1231, Feb 2014.

[29] W. Stalling, Software-Defined Networks and OpenFlow - The Internet Protocol Journal, Volume 16, No. 1, 2013.

[30] Mininet Emulator, An Instant Virtual Network on your Laptop (or other PC), http://mininet.org