

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
ESPECIALIZAÇÃO EM TELEINFORMÁTICA E REDES DE
COMPUTADORES**

JEFFERSON WANDERLEY JACOB

**PROPOSTA DE ALTA DISPONIBILIDADE PARA REDES DEFINIDAS
POR SOFTWARE ATRAVÉS DO CONTROLADOR OPENDAYLIGHT**

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA

2015

JEFFERSON WANDERLEY JACOB

**PROPOSTA DE ALTA DISPONIBILIDADE PARA REDES DEFINIDAS
POR SOFTWARE ATRAVÉS DO CONTROLADOR OPENDAYLIGHT**

Monografia de Especialização
apresentada como requisito parcial à
obtenção do título de Especialista em
Teleinformática e Redes de
Computadores, do Departamento
Acadêmico de Eletrônica da Universidade
Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Kleber Kendy
Horikawa Nabas

CURITIBA

2015

The network is the platform for
transforming the world.
(ELFRINK, Wim, 2010)

RESUMO

JACOB, Jefferson W. Proposta de alta disponibilidade para redes definidas por software através do controlador OpenDayLight. 2015. 45 f. Monografia (Especialização em Teleinformática e Redes de Computadores). Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

Este trabalho tem como objetivo implementar, em um ambiente de testes, uma rede definida por software com foco na alta disponibilidade do controlador SDN. Através da pesquisa bibliográfica busca-se apresentar os conceitos que envolvem essa arquitetura emergente, e também, as características a respeito do protocolo OpenFlow. Utilizando-se do emulador Mininet junto com o controlador OpenDayLight, implementa-se um ambiente altamente disponível, provendo a resiliência do plano de controle e garantindo a operação dos fluxos de dados.

Palavras-chave: Rede Definida por Software. OpenFlow. Emulador. OpenDayLight. Alta Disponibilidade.

ABSTRACT

JACOB, Jefferson W. High availability proposal for software-defined networks through the OpenDayLight controller. 2015. 45 f. Monografia (Especialização em Teleinformática e Redes de Computadores). Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

This work aims to implement, in a test environment, a software-defined network focused on the high availability of the SDN controller. Through a bibliographical survey, it seeks to present the concepts engaged in this emerging architecture and also the features regarding the OpenFlow protocol. Using the emulator Mininet along with the OpenDayLight controller, there is implemented a highly available environment, providing the resilience of the control plane and ensuring the operation of the data flows.

Keywords: Software Defined Network. OpenFlow. Emulator. OpenDayLight. High Availability.

LISTA DE ILUSTRAÇÕES

Figura 1 - Arquiteturas de roteadores: modelo atual (à esquerda) e modelo SDN	16
Figura 2 - Arquitetura básica SDN com um controlador externo	17
Figura 3 - Controle centralizado	18
Figura 4 - Controle distribuído	19
Figura 5 - OpenDayLight entre o protocolo aberto (OpenFlow) e a interface superior de desenvolvimento API.....	20
Figura 6 - Campos do cabeçalho de um fluxo OpenFlow.....	21
Figura 7 - Campos, ações e estatísticas de uma tabela de fluxo	22
Figura 8 - Configuração de rede no VirtualBox	27
Figura 9 - Configurações da interface de rede virtual na máquina física.....	28
Figura 10 - Topologia inicial da rede	28
Figura 11 - Diagrama da versão Hydrogen Base	29
Figura 12 - Consumo por aplicação através do comando 'top' no Linux	31
Figura 13 - Tela inicial da aplicação via interface <i>web</i>	31
Figura 14 - Resultado do comando <code>dump</code>	33
Figura 15 - Resultado do comando <code>printNodes</code> no controlador <code>c0</code>	33
Figura 16 - Interface <i>web</i> com switches OF conectados	34
Figura 17 - Menu Cluster Management.....	36
Figura 18 - Resultado do comando <code>dump</code> exibindo dois controladores.....	37
Figura 19 - Exibição do switch conectado ao controlador via interface <i>web</i>	38
Figura 20 - Acesso ao menu Flows de ambos os controladores	39
Figura 21 - Topologia de Failover.....	39
Figura 22 - Cluster Management e os respectivos nós	40
Figura 23 - Switches conectados ao <code>c1</code>	40
Figura 24 - Switches mantém conexão ao <code>c1</code>	41
Figura 25 - Controlador <code>c0</code> assume o controle dos equipamentos	41
Figura 26 - Ambos os controladores online	42

LISTA DE TABELAS

Tabela 1 - Especificações técnicas do ambiente virtual	26
--	----

LISTA DE SIGLAS

API	Application Programming Interface
CLI	Command Line Interface
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
GB	Gibabyte
Gbps	Gigabits por Segundo
GHz	Giga Hertz
GLP	General Public License
HA	High Availability
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
JVM	Java Virtual Machine
MAC	Media Access Control
MB	Megabyte
OF	OpenFlow
ONF	Open Networking Foundation
OVS	OpenvSwitch
PC	Personal Computer
RAM	Randon Access Memory
REST	Representational State Transfer
RPM	Rotações por Minuto
SDN	Software Define Network
SO	Sistema Operacional
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TSL	Transport Layer Security
VLAN	Virtual Local Area Network
VM	Virtual Machine

SUMÁRIO

1 INTRODUÇÃO	10
1.1 TEMA	10
1.1.1 Delimitação do Tema	11
1.2 PROBLEMAS E PREMISSAS	11
1.3 OBJETIVOS	12
1.3.1 Objetivo Geral	12
1.3.2 Objetivos Específicos.....	12
1.4 JUSTIFICATIVA	12
1.5 PROCEDIMENTOS METODOLÓGICOS	13
1.6 EMBASAMENTO TEÓRICO	13
1.7 ESTRUTURA	14
2 REFERENCIAIS TEÓRICOS	15
2.1 SDN	15
2.1.1 Arquitetura	15
2.1.2 Controladores	17
2.1.2.1 Controle centralizado	17
2.1.2.2 Controle distribuído	18
2.1.2.3 Controlador OpenDayLight.....	19
2.2 PROTOCOLO OPENFLOW.....	20
2.2.1 Tabela de Fluxos	21
2.2.2 Modos de Operação	23
2.2.3 Switches Openflow	24
2.2.3.1 Dedicados	24
2.2.3.2 Habilitados	24
2.2.4 Emulador Mininet.....	25
3 IMPLEMENTAÇÃO	26
3.1 DEFINIÇÃO DO AMBIENTE FÍSICO E VIRTUAL.....	26
3.2 CONTROLADOR SDN.....	29
3.3 MININET - EMULANDO SWITCHES OPENFLOW.....	32
3.4 VALIDAÇÃO	34
3.4.1 Controladores em Alta Disponibilidade	34
3.4.2 Switch OpenFlow conectado ao Cluster	36
3.4.3 Gerenciamento de Fluxos via Interface Web	38
3.4.4 Testes de Failover	39
4 CONCLUSÃO	43
5 REFERENCIAS	44

1 INTRODUÇÃO

Este capítulo apresentará elementos necessários e pertinentes ao trabalho para que possa situar o leitor com o referido tema, são eles: O tema e suas delimitações, os problemas e premissas da situação atual, seus respectivos objetivos, uma justificativa, procedimentos metodológicos adotados para o desenvolvimento do projeto, quais as bases teóricas utilizadas para a busca do conhecimento, e por fim, sua estrutura.

1.1 TEMA

A tecnologia de redes de computadores está presente em todos os níveis da sociedade, sendo que grande parte das tarefas realizada por elas – as sociedades – de alguma forma transpõe essas redes. Atualmente uma considerável parte da sociedade depende da Internet para suas atividades diárias, tanto é que ela se tornou um padrão de fácil acesso, e uma das suas características essenciais é a estabilidade. Dessa forma, inovações com novos protocolos e tecnologias ficam inviáveis, visto que o risco de interrupções das atividades para as quais ela se tornou essencial é muito grande (GUEDES et al., 2012).

Na tentativa de solucionar esse problema, pesquisadores têm apostado em iniciativas que possibilitem à implantação de redes com maiores recursos de programação, permitindo que novas tecnologias, até então descartadas, possam ser inseridas na rede gradativamente. Segundo McKeown et al. (2008) a iniciativa mais bem sucedida, sem dúvida, foi a definição da interface e do protocolo OpenFlow. Através dele os equipamentos de rede disponibilizam uma interface de programação simples, a qual permite o acesso e controle da tabela de encaminhamento utilizada pelo equipamento para determinar o próximo passo de cada pacote recebido. Neste cenário, o encaminhamento de pacotes continua sendo eficaz, pois é o hardware que ainda faz a consulta à tabela de encaminhamento, porém a decisão de como esses pacotes irão trafegar fica por conta de um controlador externo, onde diversas outras funcionalidades podem ser implementadas. É uma estrutura onde a rede pode ser programada através de software, e não fica restrita apenas ao que o

fabricante do equipamento programou, podendo ser personalizada por fornecedores ou pelos próprios operadores de rede (ROTHENBERG et al., 2010). A esse novo paradigma, deu-se o nome de Redes Definidas por Software, ou então SDN (*Software Defined Networks*).

Esse controlador OpenFlow atua como um sistema operacional (SO), gerenciando e controlando as redes (ROTHENBERG et al., 2010). Ele permite uma visão centralizada das condições da rede, porém não torna-se obrigatória a sua utilização de forma centralizada como um concentrador, podendo ser implementado de forma distribuída, visando garantir a escalabilidade e disponibilidade do sistema (GUEDES et al., 2012).

1.1.1 Delimitação do Tema

O conteúdo que será abordado inicia na compreensão das redes definidas por *software* e sua utilização através de um controlador e *switches* com suporte ao protocolo OpenFlow. Em seguida, a ênfase volta-se ao controlador em busca de uma forma de torná-lo altamente disponível através das características técnicas providas pelo fabricante.

O trabalho em questão não abordará testes de desempenho utilizando o protocolo OpenFlow em uma arquitetura de alta disponibilidade.

1.2 PROBLEMAS E PREMISSAS

Visto que, arquiteturas que utilizam um controlador centralizado enfrentam desafios quanto a capacidade de se recuperar em situações adversas (ROTHENBERG et al., 2010), como então, poderá ser realizada a mitigação de falhas neste tipo de abordagem?

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Prover a alta disponibilidade do controlador baseado em uma arquitetura de rede definida por software.

1.3.2 Objetivos Específicos

- Dissertar a respeito dos conceitos de SDN e OpenFlow.
- Emular um ambiente de rede definido por software que utilize *switches* OpenFlow, permitindo administrá-los através de um controlador externo.
- Possibilitar que a administração do tráfego (fluxos) seja realizada através de uma interface *web*.
- Demonstrar a migração dos *switches* OpenFlow, entre os controladores, na ocorrência de falha do controlador em que estão conectados.

1.4 JUSTIFICATIVA

Por definição, equipamentos como *switches* e roteadores são divididos entre duas partes: plano de controle e plano de dados. O plano de dados atua encaminhando e processando os pacotes, e o plano de controle possibilita o gerenciamento deste equipamento através de uma interface.

Fabricantes, ao criarem dispositivos de rede, geralmente integram essas partes e disponibilizam ao administrador da rede apenas o controle de funções predeterminadas, impossibilitando a manipulação de pacotes específicos, por exemplo.

O conceito de SDN foca na separação do plano de controle do plano de dados, transferindo as funções de controle para um controlador externo, o qual ignora o plano de controle fornecido pelo fabricante, permitindo ao administrador da rede mais controle do que o software embutido pelo fornecedor (COMER, 2015).

Dessa forma, ao trabalhar com controladores externos, onde a configuração dos dispositivos de rede será realizada através de um equipamento único, torna-se importante a implantação de mecanismos que atuem quanto à sua tolerância a falhas.

1.5 PROCEDIMENTOS METODOLÓGICOS

Existem diversos critérios para classificação das pesquisas. Este estudo tem por finalidade uma pesquisa aplicada, pois gera aquisição de conhecimentos que serão aplicados numa situação específica. Classificando-a conforme os propósitos e, ressaltando que envolve um levantamento bibliográfico, pode-se considerar também como uma pesquisa exploratória. E por fim, ela é delineada como uma pesquisa bibliográfica, pois é elaborada com base em material já publicado e compreende de uma seção que tem o propósito de fornecer fundamentação teórica ao trabalho (GIL, 2010).

1.6 EMBASAMENTO TEÓRICO

Este trabalho tem o foco da sua bibliografia em SDN, compreendendo o protocolo OpenFlow para o uso da mesma, e no caso de um ambiente emulado, a utilização do emulador de redes Mininet. Dessa forma, alguns trabalhos recentes foram utilizados para o desenvolvimento. Comer elucida alguns conceitos da tecnologia de rede definida por *software* através do seu livro recentemente publicado em 2015 no Brasil, Rothenberg et al. complementa com a sua pesquisa desenvolvida em 2010 através do CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações). McKeown et al. apresenta alguns conceitos e o funcionamento do protocolo OpenFlow através da sua publicação em 2008 e Lantz concluí com alguns aspectos do emulador para redes definidas por software.

1.7 ESTRUTURA

Este trabalho encontra-se dividido em quatro seções. Iniciando com a Introdução e seus respectivos itens: Tema, delimitação do tema, problemas e premissas, objetivo geral e os específicos, justificativa, procedimentos metodológicos, embasamento teórico e estrutura.

A próxima seção abordará a respeito dos conceitos sobre a tecnologia de redes definidas por software, como também o protocolo OpenFlow e o emulador de redes Mininet.

Em seguida será implementado um laboratório de testes para implantação de uma solução de SDN com suporte a alta disponibilidade, demonstrando configurações específicas e validando algumas características técnicas da solução.

E por fim, a quarta e última seção apresentará a conclusão do trabalho desenvolvido, sugerindo também, possíveis trabalhos futuros acerca do tema em questão.

2 REFERENCIAIS TEÓRICOS

2.1 SDN

Nos últimos 20 anos a evolução da arquitetura de redes baseada em camadas e nos protocolos do modelo TCP/IP não tiveram uma evolução tão considerável quanto à Internet, no que diz respeito a sua abrangência e aplicações (ROTHENBERG et al., 2010). Rothenberg et al. (2010) complementa ainda que a Internet tornando-se comercial e os equipamentos de rede cada vez mais restritos quanto à customizações, resultou em um modelo pouco flexível conhecido como engessamento. Guedes et al. (2012) utiliza a expressão ‘calcificada’ (*ossified*, em inglês), e explica que refere-se ao processo de substituição das cartilagens (mais flexíveis) por ossos ao decorrer do envelhecimento dos seres vivos.

SDN define uma abordagem que separa o processamento do plano de controle do processamento do plano de dados, transferindo para um controlador externo, que pode ser por exemplo um PC, as funções de controle (COMER, 2015).

2.1.1 Arquitetura

A Figura 1 exhibe a arquitetura atual dos roteadores e a arquitetura adotada em equipamentos de uma rede definida por software:

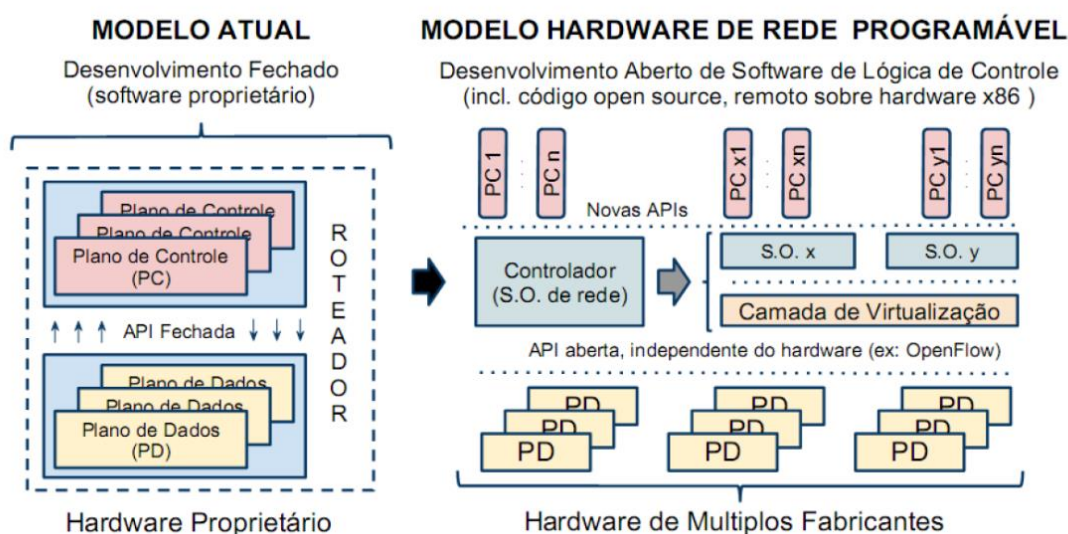


Figura 1 - Arquiteturas de roteadores: modelo atual (à esquerda) e modelo SDN
Fonte: Rothenberg et al. (2010)

É possível observar a divisão conceitual do plano de dados do plano de controle. O primeiro atua encaminhando e processando os pacotes, e o segundo é o encarregado de tomar as decisões de encaminhamento, sendo que essa configuração de encaminhamento é limitada pelo fabricante, pois o mesmo disponibiliza somente interfaces pré-determinadas de configuração (ROTHENBERG et al., 2010).

Comer (2015), através da Figura 2, ilustra a arquitetura SDN de forma mais simplista, onde um controlador viabiliza o controle externamente ao dispositivo de rede por meio de um módulo adicionado ao mesmo. Esse módulo SDN aceita comandos a nível de hardware a partir do controlador externo, e os envia um a um através da unidade de processamento de plano de dados.

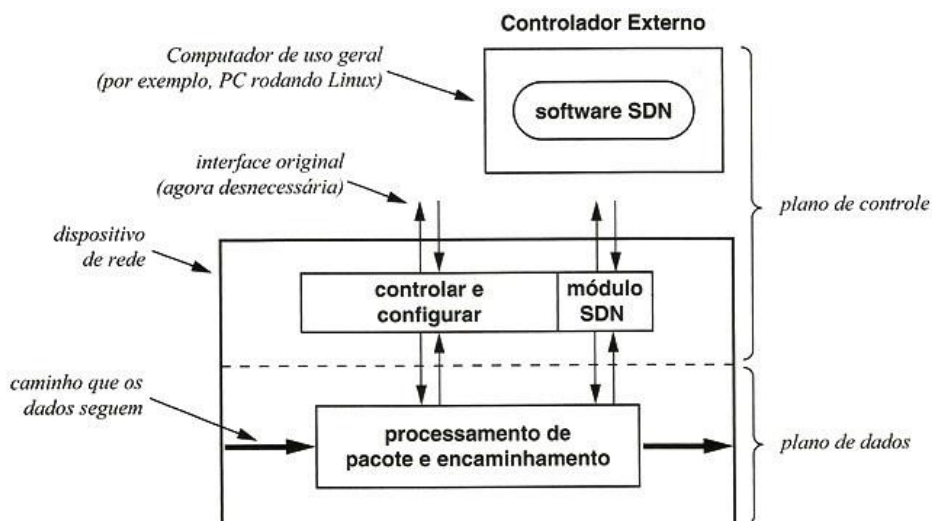


Figura 2 - Arquitetura básica SDN com um controlador externo
Fonte: Comer (2015)

É uma abordagem que permite maior controle por parte dos gerentes de rede. Através do software SDN um gerente pode configurar regras de classificação e de encaminhamento diferente daquelas permitidas pelo software do fornecedor (COMER, 2015).

2.1.2 Controladores

Controladores são responsáveis por tomar decisões de encaminhamento e gerenciar a tabela de fluxo dos equipamentos (ROTHENBERG et al., 2010). Guedes (2012) complementa: são elementos que operam como um sistema operacional para a rede, possibilitando o controle direto dos equipamentos e, disponibilizam uma interface mais eficaz aos desenvolvedores.

2.1.2.1 Controle centralizado

O controle da rede centralizado junto com uma visão global dos dispositivos possibilita a inovação e implantação de novos serviços de forma rápida por parte dos operadores de rede. Porém, ao centralizar fisicamente o controle das SDN novos desafios surgem quanto à segurança, desempenho e escalabilidade (MATTOS et al., 2015).

A Figura 3 ilustra uma arquitetura de controle centralizado onde todos os dispositivos da rede conectam ao mesmo controlador.

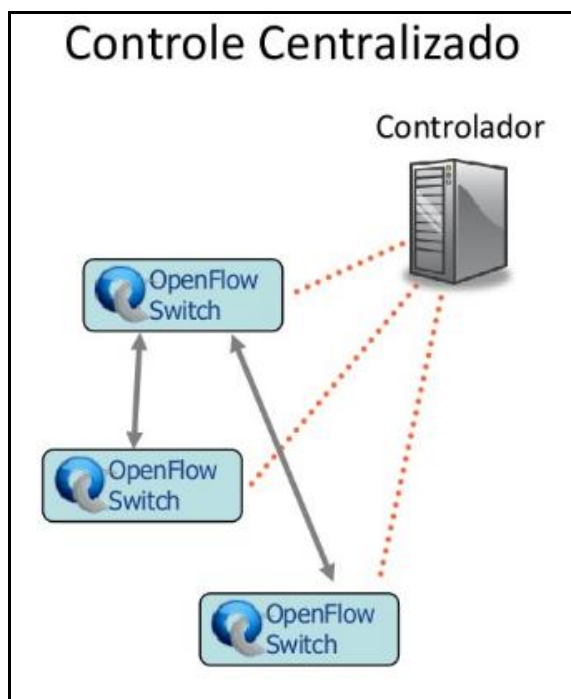


Figura 3 - Controle centralizado
Fonte: Valcy (2015)

2.1.2.2 Controle distribuído

Ao abordar a arquitetura distribuída, questões como a quantidade de controladores necessários e como devem ser interligados precisam ser avaliadas. Isso tudo está ligado diretamente ao tipo de dispositivos que eles irão controlar e também ao *software* utilizado para controle (COMER, 2015).

A Figura 4 exibe uma arquitetura utilizando controladores distribuídos onde cada equipamento de rede possui o seu próprio controlador.

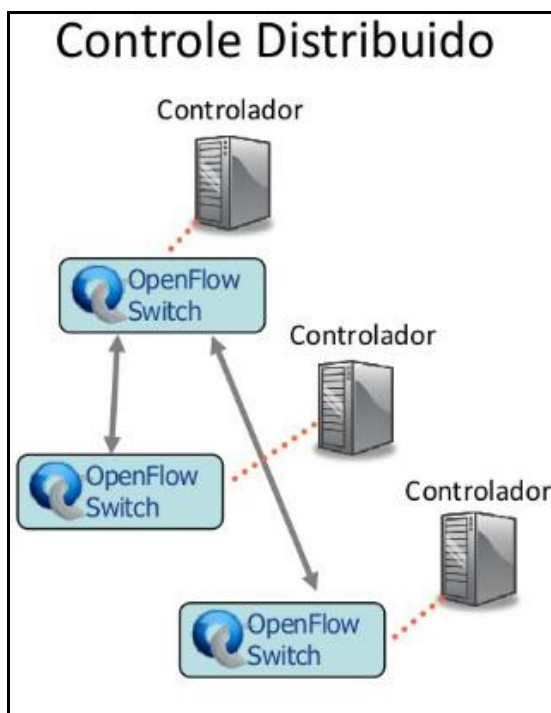


Figura 4 - Controle distribuído
Fonte: Valcy (2015)

2.1.2.3 Controlador OpenDayLight

Entre tantos projetos *open source* associados à SDN, um que se destaca na categoria de controladores é o OpenDayLight. Um projeto que tem o apoio da The Linux Foundation, cujo objetivo é acelerar a adoção do SDN por parte da indústria e dos clientes. O OpenDayLight se comunica com os equipamentos de rede através de protocolos abertos como o OpenFlow, e disponibiliza através de uma interface superior baseada em REST (*Representational State Transfer*) o controle por parte das aplicações. O ODL (OpenDayLight) fornece ainda uma interface *web* que permite a administração dos fluxos dos nós que estão sobre seu controle (NOBRE, 2014).

A Figura 5 ilustra o posicionamento do controlador em uma arquitetura SDN:

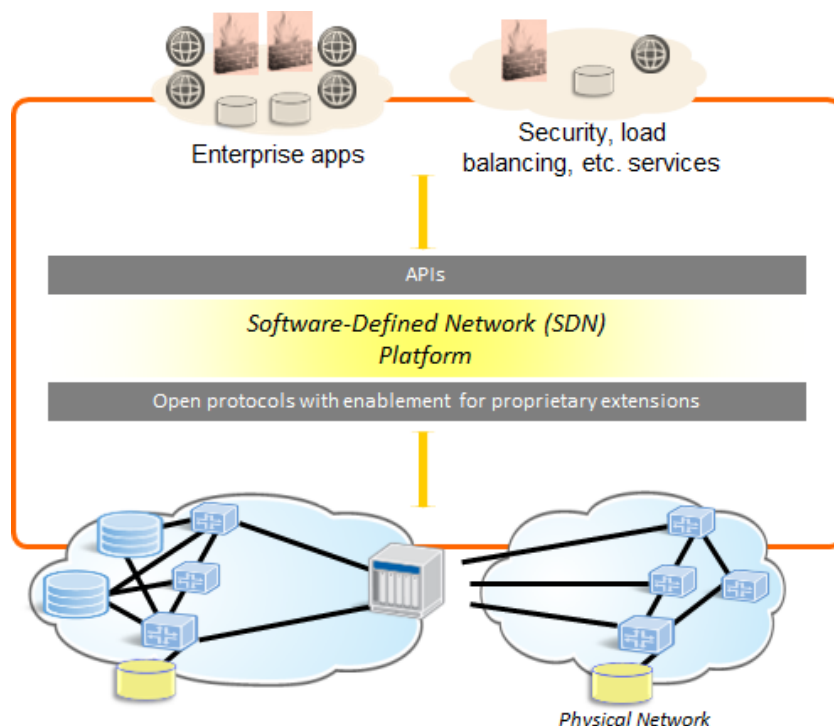


Figura 5 - OpenDayLight entre o protocolo aberto (OpenFlow) e a interface superior de desenvolvimento API
Fonte: Opendaylight.org (2015)

O projeto conta ainda com alguns parceiros tecnológicos para o seu desenvolvimento, incluindo: Brocade, Cisco, Citrix, Dell, HP, Intel, Juniper, Microsoft, NEC, Red Hat, VMWare, entre outros (OPENDAYLIGHT.ORG, 2015)

2.2 PROTOCOLO OPENFLOW

O protocolo OpenFlow foi criado na Universidade de Stanford para possibilitar aos pesquisadores realizarem experimentos com novos protocolos de rede. Diversos fabricantes adotaram o OpenFlow em seus equipamentos, e em algumas implantações de grande porte ele já está em uso. Vale lembrar que OpenFlow não é um padrão especificado pelo IETF (Internet Engineering Task Force), mas sim pelo Consórcio OpenFlow, conhecido como Open Networking Foundation (ONF) (COMER, 2015).

Em equipamentos como roteadores e *switches* clássicos, o encaminhamento de pacotes (*data path*) e a decisão de roteamento de alto nível (*control path*)

ocorrem no mesmo equipamento. Um Switch OpenFlow separa essas duas funções. A parte que pertence ao plano de dados continua no *switch*, enquanto as decisões de roteamento de alto nível são transferidas para um controlador separado, normalmente um servidor padrão. O Switch OpenFlow e o Controlador se comunicam através do protocolo OpenFlow, o qual define mensagens como: pacotes recebidos, pacotes enviados, modificar tabela de encaminhamento e gerar estatísticas.

O plano de dados (data path) de um Switch OpenFlow apresenta uma abstração clara da tabela de fluxo; cada entrada da tabela possui um conjunto de campos para validar a uma ação (como *send-out-port*, *modify-field*, ou *drop*). Quando um Switch OpenFlow recebe um pacote que nunca viu antes, e não possui nenhuma entrada de fluxo que corresponda, ele envia esse pacote ao controlador. O controlador, então, toma uma decisão em cima desse pacote, podendo descartá-lo ou, adiciona uma nova entrada da tabela de fluxos do *switch* informando como ele pode encaminhar os pacotes semelhantes a este no futuro (OPENFLOW.ORG, 2011).

2.2.1 Tabela de Fluxos

Uma das grandes vantagens do OpenFlow é a versatilidade no tratamento de cada fluxo quanto à sua ação pela rede. Cada ação é programada a partir do plano de controle, dessa forma, os fluxos podem ser configurados conforme definido pelo controlador (GUEDES et al., 2012).

A Figura 6 exibe os campos do cabeçalho de cada fluxo OpenFlow:

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

Figura 6 - Campos do cabeçalho de um fluxo OpenFlow
Fonte: McKeown et al. (2008)

Quando um pacote chega ao *switch* OF (OpenFlow) o seu cabeçalho é comparado com cada entrada da tabela de fluxos, caso ocorra uma correspondência

entende-se que o pacote corresponde aquele fluxo e determinada ação é executada, essas ações são classificadas por McKeown et al. (2008) como:

1. Encaminhar os pacotes do fluxo para determinada porta, ou um conjunto delas, permitindo assim que os pacotes sejam roteados pela rede.
2. Encapsular e encaminhar os pacotes do fluxo para um controlador através de um canal seguro (SSL/TSL). Geralmente utilizado para o primeiro pacote de um novo fluxo, permitindo ao controlador decidir se o fluxo deverá ou não ser adicionado à tabela de fluxos.
3. Descartar os pacotes do fluxo. Pode ser utilizado para bloquear ataques de negação de serviço ou *broadcast* excessivo de um determinado *host*.

Além das ações, cada entrada da tabela de fluxo possui três campos: (1) Definição do fluxo através do cabeçalho do pacote, (2) Ação de como o pacote será processado, conforme mencionado anteriormente, e (3) Geração de estatísticas para contabilizar a quantidade de pacotes e *bytes* trafegados de cada fluxo e também o tempo decorrido desde o último pacote daquele fluxo, auxiliando na remoção dos mesmos (MCKEOWN et al., 2008).

A Figura 7 ilustra as características de uma tabela de fluxo:

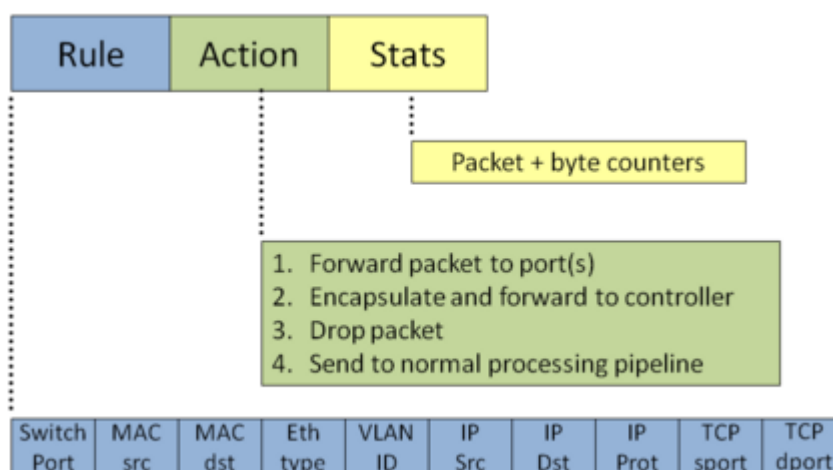


Figura 7 - Campos, ações e estatísticas de uma tabela de fluxo
 Fonte: Stanford.edu (2010)

2.2.2 Modos de Operação

Ao início de um fluxo o controlador precisar tomar as decisões quanto ao seu processamento. De acordo com Bertholdo (2012) uma rede com o OpenFlow pode operar de dois modos: reativo ou proativo.

Reativo:

- O primeiro pacote dispara a inserção do fluxo no controlador: O primeiro pacote de cada fluxo é verificado pelo controlador, depois de definido qual ação deverá ser tomada, os próximos são executados automaticamente pelo *switch*.
- Uso eficiente da tabela de fluxos: Cada regra de fluxo que é adicionada à tabela possui um *time-out*, quando esse tempo é atingido a mesma é eliminada, evitando uma sobrecarga da tabela.
- Cada fluxo tem um *delay* adicional para configuração: Atraso decorrente do primeiro pacote de um novo fluxo que deve passar pelo controlador.
- Se perde a conexão de controle, o comutador tem uma utilidade limitada: Somente fluxos que possuem regras já estabelecidas no *switch* é que irão trafegar.
- A perda da conexão de controle interrompe o tráfego: Devido ao *time-out* de cada regra, fazendo com que novos fluxos não cheguem ao controlador.

Proativo:

- O Controlador popula previamente a tabela de fluxos: Regras são adicionadas antes mesmo do primeiro pacote chegar.
- Não há perda de tempo configurando a tabela: Visto que a regra de determinado fluxo já consta na tabela, o *delay* do primeiro pacote é eliminado.

- Requer regras essencialmente agregadas (uso de coringas): Utiliza-se de campos e valores específicos para adicionar regras que compreendem uma grande quantidade de fluxos.

2.2.3 Switches Openflow

De acordo com o OpenFlow Switch Specification (2011) os *switches* OpenFlow podem ser classificados em dois tipos: Dedicados (*OpenFlow-only*) ou Habilitados (*OpenFlow-hybrid*).

2.2.3.1 Dedicados

Switches dedicados suportam apenas operações OpenFlow, ou seja, nestes equipamentos todos os pacotes são processados de acordo com as regras que compõe a tabela de fluxos, conforme explicado no item 2.2.1, e não podem operar de outra forma. Esses fluxos podem ser definidos como todos os pacotes oriundos de um mesmo endereço MAC, IP ou que possuem a mesma identificação de VLAN, por exemplo. Até mesmo pacotes não IPv4 podem ser processados, de forma experimental, através dos cabeçalhos específicos do OF (MCKEOWN et al., 2008).

2.2.3.2 Habilitados

São equipamentos comerciais como *switches*, roteadores e pontos de acesso sem fio que possuem a função de OpenFlow habilitada, à ele é adicionado uma tabela de fluxos, um canal seguro para comunicação com o controlador externo, e o protocolo OpenFlow.

Neste cenário todas as tabelas de fluxo são controladas pelo mesmo controlador, o protocolo OpenFlow permite que um *switch* possa ser controlado por dois ou mais controladores visando aumento de performance e robustez.

Um dos principais objetivos dos equipamentos com OpenFlow habilitado é possibilitar o isolamento do tráfego experimental (processado pela tabela de fluxos) do tráfego de produção, que normalmente é processado em camada 2 e 3 do *switch*. Existem duas formas de isto ser feito, a primeira é adicionar uma quarta regra no

campo de ação, fazendo com que todo pacote seja enviado para o processamento normal do *switch* (*4-Send to normal processing pipeline*), e a outra através de VLAN, utilizando identificadores para cada tipo de tráfego.

Ambas as abordagens permitem o tráfego normal de produção, o que não é experimental é processado de forma padrão pelo *switch*. Todos os *switches* OpenFlow habilitados precisam suportar uma abordagem ou outra, alguns suportam as duas (MCKEOWN et al., 2008).

2.2.4 Emulador Mininet

Mininet é um emulador de redes que permite criar redes virtuais com *hosts*, *switches*, controladores e seus respectivos links de comunicação. Seus *hosts* executam um *software* de rede padrão Linux, e seus *switches* suportam OpenFlow para roteamento flexível em redes definidas por software.

Mininet permite pesquisas, desenvolvimento, estudos, prototipação, testes, depuração e qualquer outra tarefa que pode ser benéfica quando se tem um ambiente de rede experimental em um *laptop* ou PC (MININET.ORG, 2015).

Os *switches* virtualizados no Mininet com suporte ao OpenFlow são denominados OpenvSwitch, e podem ser executados tanto em *user space* quanto em *kernel space* do Linux, obtendo melhor performance quando vinculado ao *kernel space* (LANTZ et al., 2010).

De acordo com Lantz et al. (2010), Mininet já foi utilizado por mais de 100 pesquisadores em mais de 18 instituições de ensino, como Princeton, Berkeley, Purdue, ICSI, UMass, Universidade do Alabama Huntsville, NEC, NASA, Deutsche Telekom Labs, Standford, bem como também em mais sete universidades do Brasil.

3 IMPLEMENTAÇÃO

Nesta etapa detalha-se quais foram os procedimentos adotados para construção de um laboratório virtual capaz de simular a proposta do trabalho.

3.1 DEFINIÇÃO DO AMBIENTE FÍSICO E VIRTUAL

O desenvolvimento da proposta ocorrerá através de um ambiente com três máquinas virtuais hospedadas em uma mesma máquina física, ambas em um segmento de rede isolado. A máquina física é um *notebook* com a seguinte configuração:

- Processador Intel Core i5 2.40 GHz
- 4GB de memória RAM
- Disco rígido com velocidade de 5400 RPM e 500GB de capacidade
- Interface de rede 100/1000 Gbps
- Sistema operacional Windows 8.1

Neste ambiente físico será instalado o *software* VirtualBox, que é um gerenciador de máquinas virtuais gratuito e de código aberto baseado no licenciamento GLP (VIRTUALBOX.ORG, 2015). As três máquinas virtuais hospedadas neste ambiente físico são elencadas abaixo e, utilizam a seguinte configuração:

Tabela 1 - Especificações técnicas do ambiente virtual

Hardware	Controlador SDN Primário	Controlador SDN Secundário	Emulador de switches OpenFlow
Processador	02 Core 2.4GHz	02 Core 2.4GHz	02 Core 2.4GHz
Memória	1024 MB	1024 MB	128 MB
Disco	10 GB	10 GB	8 GB
Sistema Operacional	Ubuntu 12.04	Ubuntu 12.04	Ubuntu 14.04
Endereço IP	192.168.0.100	192.168.0.200	192.168.0.1

Fonte: Autoria própria

Para que essas máquinas virtuais comuniquem-se entre si, em uma rede isolada, será necessário ajustar a configuração da interface de cada uma delas apontando para uma placa de rede exclusiva do hospedeiro, conforme Figura 8 a seguir:

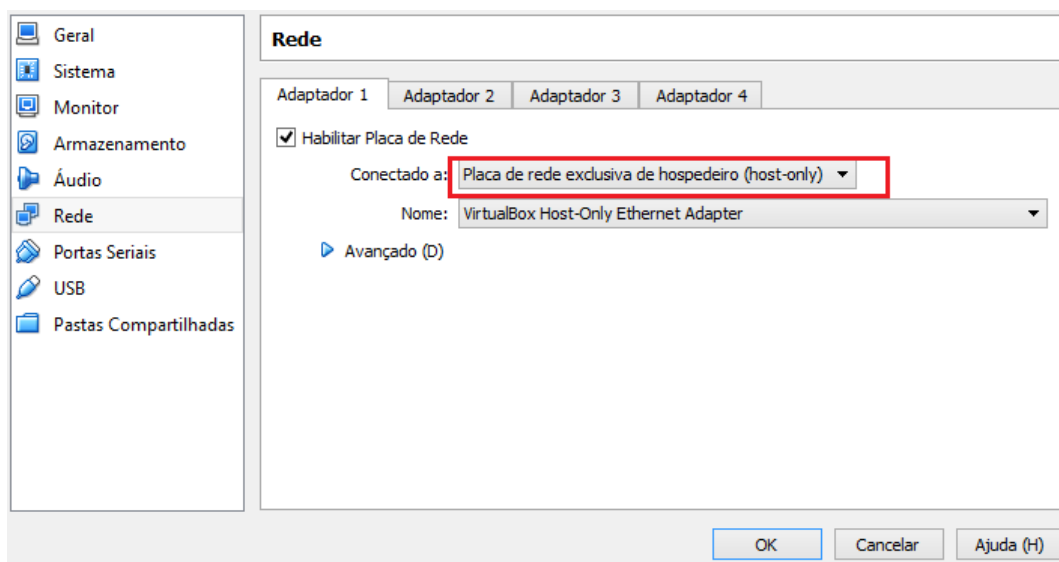


Figura 8 - Configuração de rede no VirtualBox
Fonte: Autoria própria

Dessa forma, as máquinas virtuais além de se comunicarem entre si também se comunicarão com o ambiente físico onde estão hospedadas, isso será necessário, pois a partir deste ambiente físico que o acesso aos respectivos ambientes virtuais será realizado.

A figura a seguir ilustra a saída do comando `ipconfig /all` da respectiva interface no computador que hospeda as máquinas virtuais, é através dela que será realizado o acesso aos ambientes:

```

Prompt de Comando

Adaptador Ethernet VirtualBox Host-Only Network:

Sufixo DNS específico de conexão . . . . . :
Descrição . . . . . : VirtualBox Host-Only Ethernet Adapter
Endereço Físico . . . . . : 08-00-27-00-BC-98
DHCP Habilitado . . . . . : Não
Configuração Automática Habilitada . . . . . : Sim
Endereço IPv6 de link local . . . . . : fe80::f9fe:7087:d4cc:9347%11<Preferencial>
Endereço IPv4 . . . . . : 192.168.0.50<Preferencial>
Máscara de Sub-rede . . . . . : 255.255.255.0
Gateway Padrão . . . . . :
IÁID de DHCPv6 . . . . . : 403177511
DUID de Cliente DHCPv6 . . . . . : 00-01-00-01-1C-1B-9B-FC-E0-DB-55-FF-C5-41
Servidores DNS . . . . . : fec0:0:0:ffff::1%1
                          fec0:0:0:ffff::2%1
                          fec0:0:0:ffff::3%1

NetBIOS em Tcpip. . . . . : Habilitado

```

Figura 9 - Configurações da interface de rede virtual na máquina física
Fonte: Autoria própria

A imagem abaixo, ilustra de forma resumida, a topologia da rede de como ficou essa definição.

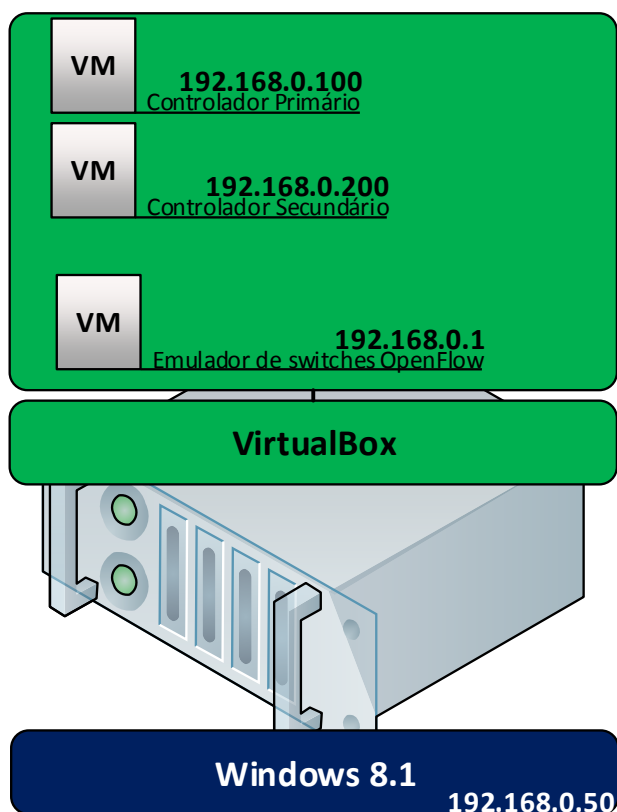


Figura 10 - Topologia inicial da rede
Fonte: Autoria própria

3.2 CONTROLADOR SDN

A escolha do controlador OpenDayLight ocorreu devido a dois motivos principais, um deles é a possibilidade de administrá-lo através de uma interface *web* e, o outro é o fato de que a existe um grande projeto sendo realizado por várias empresas que apostam na tecnologia SDN, conforme foi mencionado no item 2.1.2.3.

Após alguns testes iniciais para definição da versão, optou-se por utilizar a que comumente é conhecida como Hydrogen, pois a versão Helium, que é a mais recente, apresentou alguns problemas bem significantes quanto à proposta do trabalho, muitos deles são até questionados no fórum oficial e outros são corrigidos através de uma forma não convencional.

Abaixo, na Figura 11, um diagrama da versão que será utilizada:

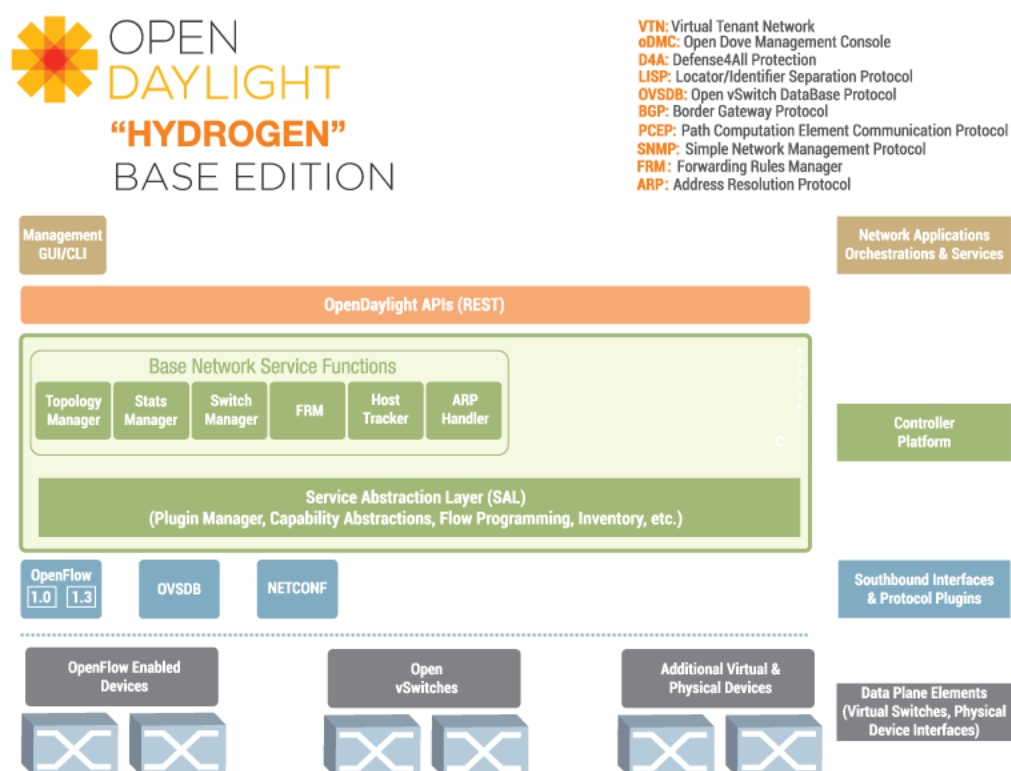


Figura 11 - Diagrama da versão Hydrogen Base
 Fonte: Opendaylight.org (2015)

Para configurá-lo no ambiente Linux será necessário primeiro a instalação de uma Máquina Virtual Java (JVM) e definir a variável JAVA_HOME para o diretório

desta instalação. Para prosseguir com a instalação da JVM utilizou-se o seguinte comando:

```
apt-get install openjdk-7-jdk
```

Em seguida é definida a variável. Para isso, deve-se editar o arquivo `/etc/environment` e adicionar a seguinte linha:

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Após isso, o arquivo deverá ser recarregado através do seguinte comando:

```
source /etc/environment
```

Em seguida, efetua-se um teste através do comando `echo` para validar a variável de ambiente. A saída do comando deverá retornar o diretório acima, conforme abaixo:

```
echo $JAVA_HOME  
/usr/lib/jvm/java-7-openjdk-amd64
```

Após concluído os pré-requisitos do Installation Guide, deve-se prosseguir com o *download* da aplicação e, em seguida, a sua execução.

A aplicação é executada em uma JVM e por isso não é necessário realizar algum tipo de compilação. Deve-se então baixar e descompactar o arquivo do repositório oficial através dos comandos `wget` e `unzip -a`, conforme abaixo:

```
wget  
https://nexus.opendaylight.org/content/repositories/opendaylight.release/org.opendaylight/integration/distributions-base/0.1.1/distributions-base-0.1.1-osgipackage.zip  
  
unzip -a distributions-base-0.1.1-osgipackage.zip
```

Após a descompactação, um novo diretório com o nome `/opendaylight` é criado, nele estão todos os arquivos para a execução do controlador SDN. Para iniciá-lo é preciso executar o arquivo `run.sh` através do comando `./run.sh` com privilégios de administrador e aguardar a inicialização. Nota-se, na figura abaixo, que após a execução, o controlador consome praticamente 70% da memória disponível de uma máquina com apenas 1GB:

```

top - 16:10:14 up 3:15, 3 users, load average: 0.00, 0.04, 0.10
Tasks: 85 total, 1 running, 84 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.3%us, 2.0%sy, 0.0%ni, 96.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1017436k total, 928240k used, 89196k free, 8368k buffers
Swap: 1987580k total, 2960k used, 1984620k free, 123508k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 1629 root        20   0 1991m 691m 15m  S   6   69.6    2:41.89 java

```

Figura 12 - Consumo por aplicação através do comando 'top' no Linux
Fonte: Autoria própria

Foram realizadas algumas tentativas de executá-lo com menos de 1GB de memória, mas em todos os teste a inicialização ocorria de forma muito lenta e em alguns momentos a aplicação finalizava automaticamente, dessa forma optou-se por utilizar 1GB conforme a recomendação mínima do Installation Guide (OPENDAYLIGHT.ORG, 2014).

A partir deste momento já é possível acessá-lo através de sua interface *web* utilizando a porta 8080. A seguir, a Figura 13 exhibe a tela inicial após autenticação com o usuário e senha *admin*:

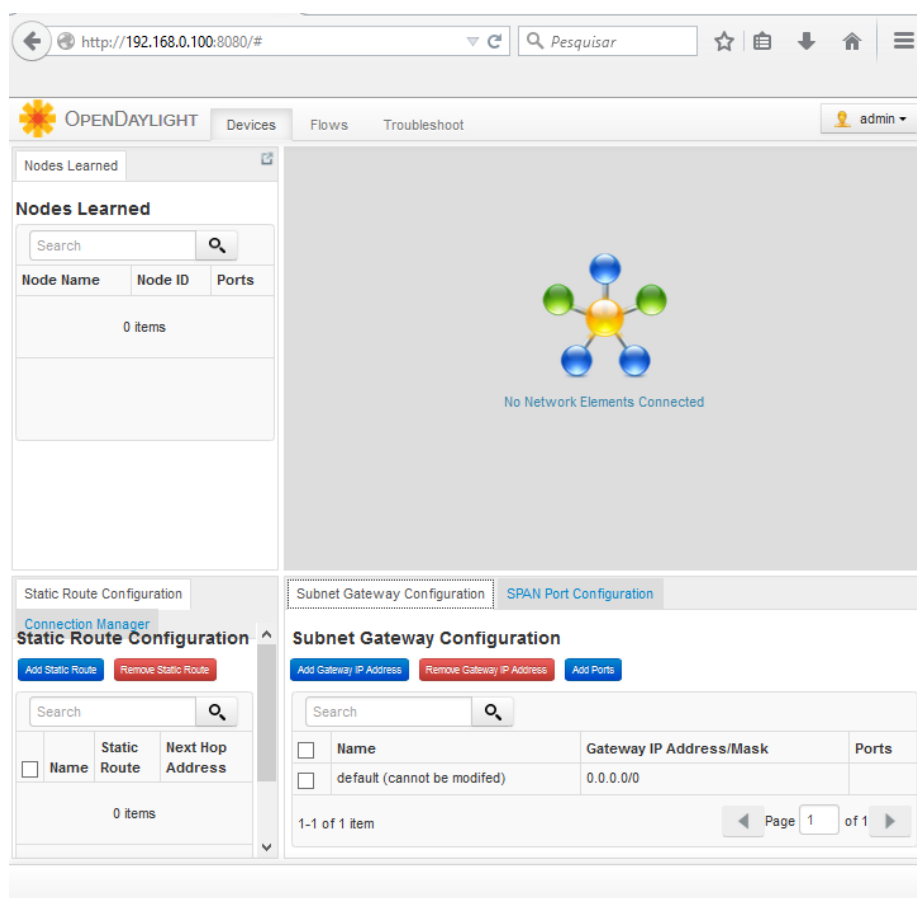


Figura 13 - Tela inicial da aplicação via interface *web*
Fonte: Autoria própria

Neste primeiro momento ainda não existem *switches* OpenFlow conectados ao controlador e, conseqüentemente também não há fluxos e nem regras na tabela de fluxos.

3.3 MININET - EMULANDO SWITCHES OPENFLOW

Para possibilitar o uso de *switches* OpenFlow, será necessário utilizar um emulador, neste caso optou-se pelo Mininet, que é capaz emular uma rede SDN completa, conforme mencionado no item 2.2.4. Neste trabalho ele é utilizado apenas para *switches*, *hosts* e *links*.

O site oficial da ferramenta disponibiliza uma máquina virtual pronta para executá-la no VirtualBox, será utilizada a versão 2.2.1, que pode ser obtida através do link abaixo:

```
http://downloads.mininet.org/mininet-2.2.1-150420-ubuntu-14.04-server-i386.zip
```

Optou-se pela versão i386 (32 bits) para possibilitar reduzir ao máximo a quantidade de memória utilizada pela VM, visto que, existe uma limitação de 4GB para todo o ambiente, físico e virtual.

Após o download desta máquina virtual, é necessário ajustar suas configurações de rede no VirtualBox para *host-only*, conforme mencionado no item 3.1. Além disso, é preciso também ajustar seu endereçamento IP no Linux, o qual poderá ser realizado através do arquivo `/etc/network/interfaces`, ficando conforme abaixo:

```
auto eth0
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
```

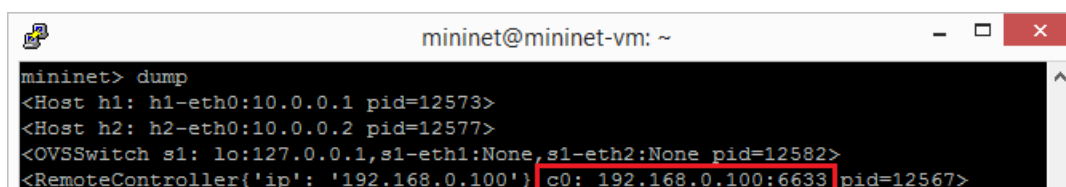
Após essas etapas já é possível conectar *switches* OpenFlow ao controlador OpenDayLight através do seguinte comando:

```
sudo mn --controller=remote,ip=192.168.0.100 --topo=minimal
```


Onde:

- `mn`: Executa aplicação mininet;
- `--controller=remote,ip`: Define se o controlador utilizado será remoto ou local;
- `--topo`: Define a topologia da rede escolhida;

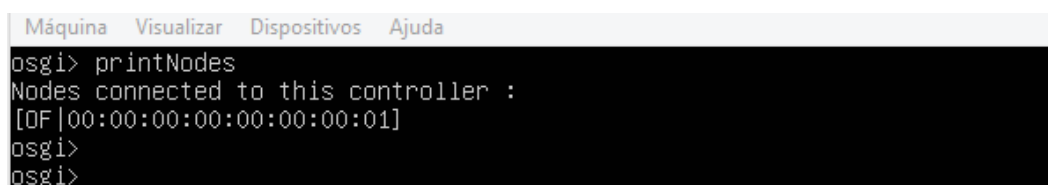
Neste exemplo os *switches* são conectados em apenas um controlador e ainda não há alta disponibilidade. Através do comando `dump`, com o Mininet ainda em execução conforme a Figura 14, é possível identificar o controlador como `c0`:



```
mininet@mininet-vm: ~
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=12573>
<Host h2: h2-eth0:10.0.0.2 pid=12577>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=12582>
<RemoteController{'ip': '192.168.0.100'} c0: 192.168.0.100:6633 pid=12567>
```

Figura 14 - Resultado do comando dump
Fonte: Autoria própria

E através do comando `printNodes`, no console do controlador, é possível obter quais os *switches* estão conectados ao `c0`, conforme ilustra a figura abaixo:



```
Máquina Visualizar Dispositivos Ajuda
osgi> printNodes
Nodes connected to this controller :
[0F|00:00:00:00:00:00:01]
osgi>
osgi>
```

Figura 15 - Resultado do comando printNodes no controlador c0
Fonte: Autoria própria

A partir deste momento já é possível observar, através da interface *web*, a exibição de uma topologia e também uma lista de *switches* que estão conectados ao controlador, conforme ilustra a Figura 16 a seguir:

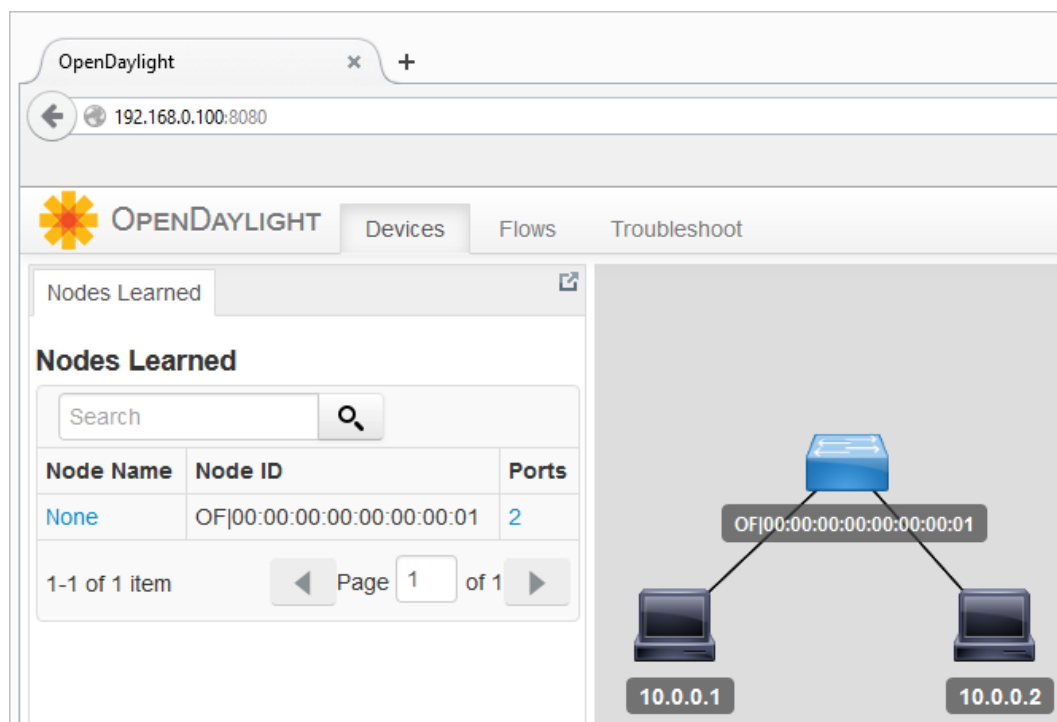


Figura 16 - Interface web com switches OF conectados
Fonte: Aatoria própria

3.4 VALIDAÇÃO

Na etapa anterior foi demonstrado como foram realizadas as configurações iniciais e necessárias para trabalhar em um ambiente de rede definido por *software*, incluindo também alguns comandos para evidenciar determinadas características técnicas.

Nesta etapa demonstra-se como tornar esse ambiente altamente disponível, tornando-o assim, mais seguro e confiável.

3.4.1 Controladores em Alta Disponibilidade

Para configurar um *cluster* entre os controladores deve-se iniciar a aplicação através do comando `-Dsupernodes` informando quais são os nós que irão fazer parte, neste caso, quem irá executar o controlador em conjunto com o seu par. Além disso, dois modos de operação são suportados e configurados através do seguinte arquivo (OPENDAYLIGHT.ORG, 2013):

/opendaylight/configuration/config.ini

São eles:

- SINGLE_CONTROLLER: Todos os dispositivos (*switches*) serão conectados a apenas um controlador.
- ANY_CONTROLLER_ONE_MASTER: Qualquer dispositivo pode conectar em qualquer controlador. Mas somente um é o *Master* (principal).

Por padrão a opção utilizada é ANY_CONTROLLER_ONE_MASTER, deve-se mantê-la, pois atendeu bem aos testes realizados. O arquivo de configuração deverá então manter-se inalterado, ficando da seguinte forma em ambos os controladores:

```
# Connection manager configuration
connection.scheme = ANY_CONTROLLER_ONE_MASTER
```

A partir desse momento, já é possível iniciar os controladores através da CLI de cada um deles, utilizando o comando abaixo a partir do diretório da aplicação:

```
./run.sh -Dsupernodes=192.168.0.100:192.168.0.200
```

Após o processo de inicialização da aplicação, pode-se validar quais são os nós que fazem parte do *cluster* utilizando o comando abaixo:

```
osgi> getClusterNodes
      192.168.0.100
      192.168.0.200
```

Em ambos os controladores o resultado aparece idêntico ao acima.

Uma outra validação, muito similar, é através da interface *web*, acessando o item *Admin>Cluster* através do menu superior direito, conforme mostra a imagem abaixo:

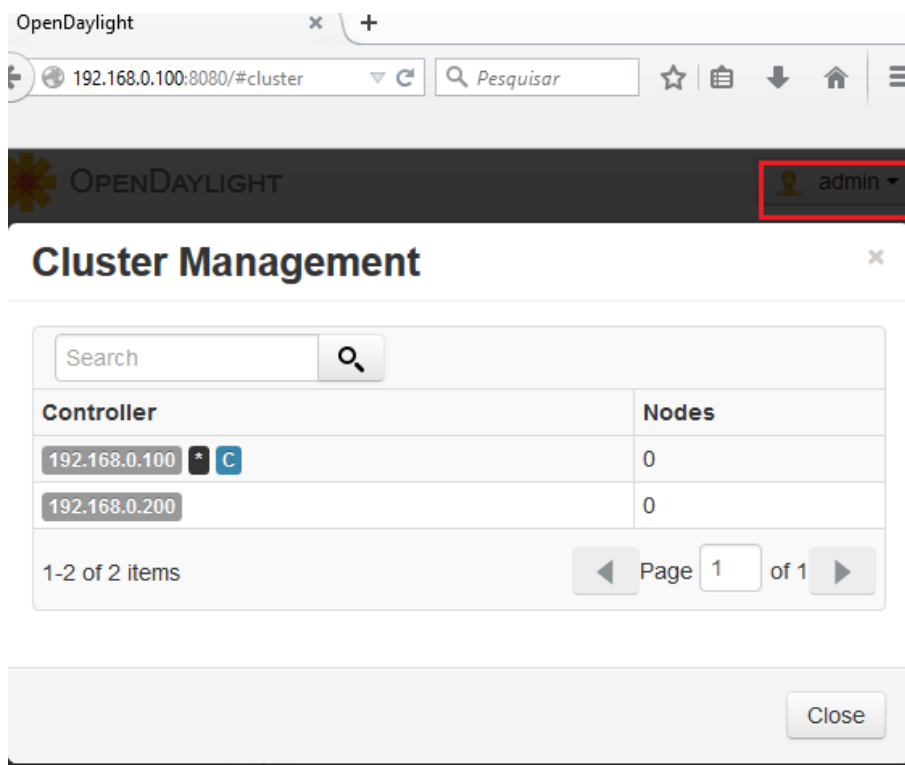


Figura 17 - Menu Cluster Management
Fonte: Autoria própria

Neste caso, o controlador com IP 192.168.0.100 assumiu como o principal, o sinal de * significa que é através dele que está sendo realizado o acesso ao menu, conforme identificado também na barra de endereços do navegador e, no momento ambos não possuem *switches* (*Nodes*) conectados.

3.4.2 Switch OpenFlow conectado ao Cluster

Tendo em vista que o ambiente de gerência encontra-se altamente disponível, é preciso informar aos *switches* que irão fazer parte desta rede, quais são os controladores em que eles irão se conectar, pois dessa forma por exemplo, a falha de um controlador não afetaria o gerenciamento destes *switches*.

Similar ao comando já executado no item 3.3, porém, agora com adição do controlador secundário na mesma linha de execução do Mininet, o comando que irá executar a aplicação que emula *switches* OpenFlow ficará da seguinte forma:

```
sudo mn --controller=remote,ip=192.168.0.100 --
controller=remote,ip=192.168.0.200 --topo=minimal
```

Neste caso, é informado ao *switch* o endereço IP de todos os nós que compõe o *cluster*, permitindo ao equipamento comunicar-se com qualquer um deles.

Através do comando `dump` no Mininet, pode-se identificar que a rede possui dois controladores, `c0` e `c1`, conforme a Figura 18:

```

mininet@mini
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=22565>
<Host h2: h2-eth0:10.0.0.2 pid=22569>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=22574>
<RemoteController{'ip': '192.168.0.100'} c0: 192.168.0.100:6633 pid=22553>
<RemoteController{'ip': '192.168.0.200'} c1: 192.168.0.200:6633 pid=22559>

```

Figura 18 - Resultado do comando `dump` exibindo dois controladores
 Fonte: Autoria própria

É possível ainda validar essa conexão através de um comando específico que estabelece comunicação com *switches* que suportam OpenFlow, como é o caso do OpenvSwitch (OVS), o qual já está incluso na máquina virtual do Mininet. Através dele é possível obter informações específicas dos *switches*. O comando utilizado é `sudo ovs-vsctl show`, ele será executado através de um outro terminal SSH e trará o seguinte resultado:

```

mininet@mininet-vm:~$ sudo ovs-vsctl show
aaa4c93f-aa5a-4e41-b565-47c6f100c291
    Bridge "s1"
        Controller "tcp:192.168.0.100:6633"
            is_connected: true
        Controller "ptcp:6634"
        Controller "tcp:192.168.0.200:6633"
            is_connected: true
        fail_mode: secure
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth1"
            Interface "s1-eth1"
    ovs_version: "2.0.2"

```

O resultado exibe um único *switch* da topologia definido como `s1`, e logo abaixo vem os controladores em que está conectado, entre outras informações.

Também é possível verificar essas informações através da interface *web* do controlador. Novamente, através do Cluster Management, pode-se visualizar a quantidade de *switches* e também qual dos nós o controlador elegeu como *Master* para aquele determinado equipamento, conforme abaixo:

Controller	Nodes
192.168.0.100 * C	1
192.168.0.200	0

1-2 of 2 items Page 1 of 1

Figura 19 - Exibição do switch conectado ao controlador via interface web
 Fonte: Autoria própria

Além disso, o comando `printNodes`, demonstrado no item 3.3, também poderá ser executado para consultar essa informação.

3.4.3 Gerenciamento de Fluxos via Interface Web

Baseado em um dos objetivos específicos, conforme especificado na Introdução, a gestão de fluxos deste *switch* também é realizada via interface *web*. Ao acessar o controlador é possível identificar o menu Flows, através dele pode-se realizar a inclusão e exclusão destes fluxos, independente por qual controlador é realizado o acesso, obtendo dessa forma um gerenciamento distribuído.

A Figura 20 ilustra o acesso sendo realizado a partir dos dois controladores. Inicialmente a regra de fluxo com o nome *droptest* foi configurada a partir do controlador com IP 192.168.0.100, mas como o ambiente em questão possui as características de alta disponibilidade, essa mesma regra pode então ser gerenciada a partir de controlador com IP 192.168.0.200 também, ou qualquer outro que faça parte do *cluster*.

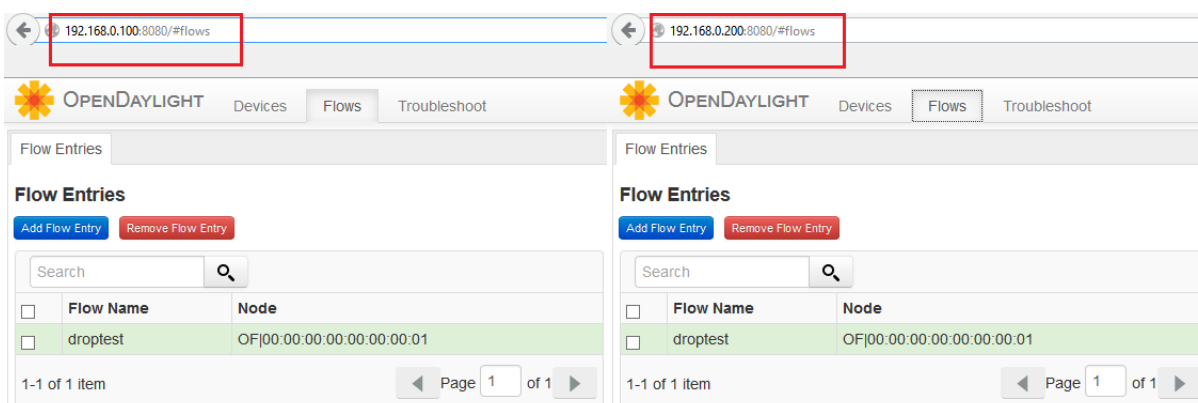


Figura 20 - Acesso ao menu Flows de ambos os controladores
Fonte: Autoria própria

3.4.4 Testes de Failover

A topologia de rede a seguir, Figura 21, ilustra o ambiente virtual e procura facilitar o entendimento dos testes que serão realizados. Essa figura define a topologia escolhida, utilizando dois controladores denominados c0 e c1, e também três *switches*, ambos conectados aos controladores em questão.

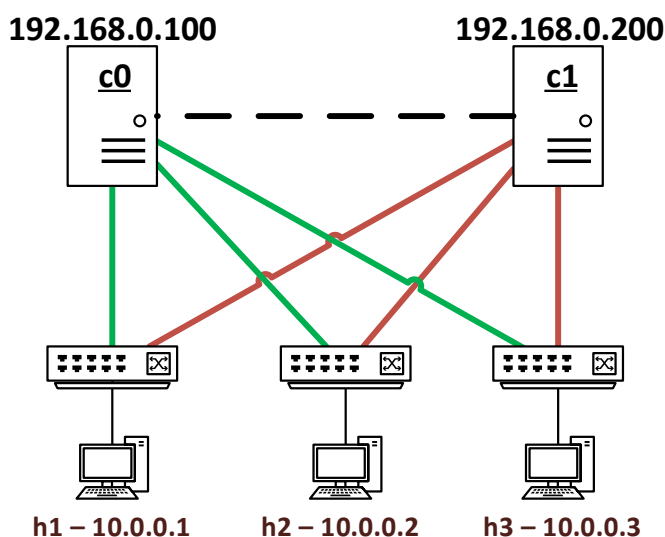


Figura 21 - Topologia de Failover
Fonte: Autoria própria

Dessa vez a topologia será criada com mais de um *switch*, visando demonstrar com mais clareza os resultados.

Assim a topologia acima é criada através do Mininet com o seguinte comando:

```
sudo mn --controller=remote,ip=192.168.0.100 --
controller=remote,ip=192.168.0.200 --topo=linear,3
```

Pode-se observar que neste primeiro momento nem todos os *switches* estão com a comunicação preferencial no mesmo controlador, o que caracteriza o modo de operação ANY_CONTROLLER_ONE_MASTER. A Figura 22 ilustra essa característica.

Controller	Nodes
192.168.0.100 * C	2
192.168.0.200	1

1-2 of 2 items Page 1 of 1

Figura 22 - Cluster Management e os respectivos nós
Fonte: Autoria própria

Em seguida é realizada uma simulação de falha no controlador c0, que possui IP final 100, para isso a execução do ODL neste ambiente será interrompida, fazendo com que os *switches* automaticamente migrem para o controlador c1, que possui IP final 200. A saída do comando `printNodes` executado no c1 após a falha evidencia essa informação, mostrando onde os *switches* estão conectados:

```
osgi> printnodes
Nodes connected to this controller :
[OF|00:00:00:00:00:00:00:02,          OF|00:00:00:00:00:00:00:03,
OF|00:00:00:00:00:00:00:01]
```

Além desse comando, o próprio Cluster Management também apresenta informações de onde os *switches* estão conectados naquele momento:

Controller	Nodes
192.168.0.200 * C	3

1-1 of 1 item Page 1 of 1

Figura 23 - Switches conectados ao c1
Fonte: Autoria própria

Nota-se que a partir do momento em que um nó do *cluster* falha, ele já não é mais apresentado junto com os outros nós que compõem a sua estrutura, porém,

logo que o comando de inicialização do controlador é executado novamente, o mesmo volta a integrar a solução automaticamente.

A figura abaixo demonstra essa característica. Neste momento os dois controladores já estão disponíveis, e detalhe, mesmo com o retorno do controlador c0 os *switches* permanecem conectados em c1, caracterizando novamente o modo de operação ANY_CONTROLLER_ONE_MASTER.

192.168.0.200 C	3
192.168.0.100 *	0

1-2 of 2 items Page 1 of 1

Figura 24 - Switches mantém conexão ao c1
Fonte: Autoria própria

De qualquer forma, o ambiente do c0 encontra-se apto a controlar os equipamentos.

Agora uma nova falha será simulada, dessa vez em c1, o controlador que aparece com todos (três) os *switches*. A Figura 25 exibe o Cluster Management após a falha do c1. Percebe-se que os *switches* que estavam com o controlador final 200 agora aparecem com o de final 100 (c0), que retornou da falha anterior e já está operando novamente.

Controller	Nodes
192.168.0.100 * C	3

1-1 of 1 item Page 1 of 1

Figura 25 - Controlador c0 assume o controle dos equipamentos
Fonte: Autoria própria

Isso comprova que independente de qual controlador falhar, desde que exista pelo menos um em funcionando, é possível manter a estrutura online. Novamente, após o retorno de c1, o ambiente volta a ficar totalmente disponível. A figura a seguir ilustra esse comportamento já conhecido.

Controller	Nodes
192.168.0.100 	3
192.168.0.200 	0

Figura 26 - Ambos os controladores online
Fonte: Autoria própria

4 CONCLUSÃO

Com o desenvolvimento deste trabalho conclui-se que é possível alcançar um dos aspectos de segurança quando se trata de controle centralizado, neste caso, a alta disponibilidade do controlador de redes definidas por software. Os conceitos abordados sobre a arquitetura SDN e o protocolo de comunicação, mais comumente conhecido como OpenFlow, foram de grande valia para a preparação do ambiente virtual. O controlador escolhido, OpenDayLight, atendeu a proposta quanto a capacidade de configurá-lo através de uma interface *web*, dispensando assim linhas de código ao se realizar operações básicas com os fluxos. Sua integração com o emulador Mininet ocorreu conforme o esperado e ambas as ferramentas colaboraram para a aquisição de novos conhecimentos a respeito de suas tecnologias.

Uma das dificuldades durante a elaboração deste trabalho foi em relação à busca pelo conteúdo bibliográfico, visto que as ferramentas utilizadas e a tecnologia empregada estão em constante evolução, e ainda não existe um padrão especificado pelo IETF.

Como proposta para trabalhos futuros, propõe-se a utilização de um serviço de balanceamento através de um IP virtual onde os *switches* OpenFlow irão se conectar, dessa forma, ao invés de especificar o endereço IP de cada controlador no momento da conexão, utiliza-se apenas um, resultando em um ambiente mais transparente, proporcionando escalabilidade a nível de controle.

5 REFERENCIAS

BERTHOLDO, Leandro. **Tecnologias, conceitos e serviços emergentes: Openflow**. 13o Workshop RNP, 2012. Disponível em: <http://www.pop-rs.com.br/images/publicacoes/2012/WRNP_Openflow.pdf>. Acesso em: 15 jul. 2015

COMER, Douglas E. **Interligação de redes com TCP/IP. Vol. 1 Princípios, protocolos e arquitetura**. Rio de Janeiro: Editora Elsevier, 2015.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Editora Atlas, 2010.

GUEDES, Dorgival et al. **Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores**. Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012, v. 30, n. 4, p. 160-210, 2012. Disponível em: <<http://www.researchgate.net/publication/260346033>>. Acesso em: 01 jul. 2015

LANTZ, Bob; HELLER, Brandon; MCKEOWN, Nick. **A network in a laptop: rapid prototyping for software-defined networks**. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010. p. 19. Disponível em: <<http://conferences.sigcomm.org/hotnets/2010/papers/a19-lantz.pdf>>. Acesso em: 16 jul. 2015

MATTOS, D. M. F., Lopez, M. E. A., Ferraz, L. H. G. e Duarte, O. C M. B. (2015). **Controlador resiliente com distribuição eficiente para redes definidas por software**. In: XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2015. Disponível em: <<http://sbrc2015.ufes.br/wp-content/uploads/138820.1.pdf>>. Acesso em: 09 jul. 2015

MCKEOWN, Nick et al. **OpenFlow: enabling innovation in campus networks**. In: ACM SIGCOMM Computer Communication Review, v. 38, n. 2, p. 69-74, 2008. Disponível em: <<http://archive.openflow.org/documents/openflow-wp-latest.pdf>>. Acesso em: 01 jul. 2015

MININET.ORG. **Mininet Overview**. 2015. Disponível em <<http://mininet.org/overview>>. Acesso em: 16 jul. 2015

NOBRE, Tito Sérgio Martins Pereira. **SDN em rede de transporte ópticas**. 2014. 107 f. Dissertação (Mestrado em Engenharia Informática) - Faculdade de Ciências, Universidade de Lisboa, 2014. Disponível em: <http://docs.di.fc.ul.pt/bitstream/10451/15936/1/ulfc112524_tm_Tito_Nobre.pdf>. Acesso em: 10 jul. 2015

OPENDAYLIGHT.ORG. **The OpenDaylight Project**. 2015. Disponível em: <http://go.linuxfoundation.org/l/6342/2015-06-10/2hcq9/6342/127298/OpenDaylight_Briefing_Deck_06102015.pptx>. Acesso em: 12 jul. 2015

_____. **First Code Release "Hydrogen" Diagram**. 2015. Disponível em: <<http://www.opendaylight.org/resources/collateral>>. Acesso em: 13 mai. 2015

_____. **OpenDaylight Controller:Programmer Guide:Clustering**. 2013. Disponível em: <https://wiki.opendaylight.org/view/OpenDaylight_Controller:Programmer_Guide:Clustering>. Acesso em: 19 mai. 2014

_____. **Release/Hydrogen/Base/Installation Guide**. 2014. Disponível em: <https://wiki.opendaylight.org/view/Release/Hydrogen/Base/Installation_Guide>. Acesso em: 13 mai. 2015

OPENFLOW.ORG. **How does OpenFlow work?**. 2011. Disponível em: <<http://archive.openflow.org/wp/learnmore>>. Acesso em 13 jul. 2015

ROTHENBERG, Christian Esteve et al. **OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes**. Cad. CPqD Tecnologia, Campinas, v. 7, n. 1, p. 65-76, 2010. Disponível em: <<http://www.researchgate.net/publication/266292305>>. Acesso em: 14 jun. 2015

STANFORD.EDU. **Flow Table Entries**. 2010. Disponível em: <<http://yuba.stanford.edu/cs244wiki/index.php/Overview>>. Acesso em: 14 jul. 2015

VALCY, Italo. **Software-Defined Networks e Openflow: conceitos e tecnologias emergentes**. III Workshop de Tecnologia de Redes do PoP-BA Ponto de Presença da RNP na Bahia, 2012. Disponível em: <<https://www.pop-ba.rnp.br/pub/WTR2012/Programacao/01-WTR2012-SDN-Openflow.pdf>>. Acesso em: 09 jul. 2015

VIRTUALBOX.ORG. **Oracle VM Virtual Box**. 2015. Disponível em: <<https://www.virtualbox.org>>. Acesso em: 12 mai. 2015