OrchFlow

Uma Ferramenta para Orquestração de
Múltiplos Controladores OpenFlow

OrchFlow

Manual de instalação e uso

UFSCAR - Universidade Federal de São Carlos
SOROCABA - 2016

LERIS - Laboratory of Studies in Networks, Innovation and Software
UFSCAR - Sorocaba
http://www.sorocaba.ufscar.br/ufscar

Título:
OrchFlow - Uma Ferramenta para Orquestração de Múltiplos Controladores OpenFlow

Evento:
Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)

Fórum:
Salão de Ferramentas do SBRC - 2016

Grupo de trabalho:
Universidade Federal de São Carlos (UFSCar) - Sorocaba

Participante(s):
Marcelo Frate
Marcelo K. M. Marczuk

Orientador(s):
Fábio L. Verdi

Cópias: 1

Numero de páginas: 24

Data de Publicação:
8 de março de 2016

Resumo:

O principal objetivo das redes definidas por software é a centralização da lógica de controle, porém é possível dividir esta lógica entre dois ou mais controladores com o intuito de garantir a escalabilidade. O protocolo OpenFlow define a comunicação entre switches e controladores, entretanto não prevê a comunicação entre controladores, necessária para qualquer tipo de distribuição no plano de controle. Faz-se necessário portanto o desenvolvimento de soluções independentes do protocolo, capazes de distribuir essa lógica dentro de um mesmo domínio administrativo. Neste cenário, o OrchFlow surge como uma ferramenta capaz de orquestrar uma rede definida por software, com dois ou mais controladores OpenFlow, permitindo a gerência e o monitoramento da topologia em tempo real.

Sumário

Prefácio **vii**

1 Introdução **1**
1.1 Arquitetura do OrchFlow 1

2 Funcionamento do OrchFlow **3**
2.1 Detalhes da implementação 3

3 Requisitos de Software **5**
3.1 Funcionalidades 5
3.2 Requisitos 5

4 Inicialização do sistema de testes **7**
4.1 Acesso ao Servidor LERIS 8

5 Acessando o sistema OrchFlow **9**

6 Utilizando o OrchFlow **13**
6.1 Ping 14
6.2 SSH 15
6.3 Servidor HTTP 16
6.4 Servidor FTP 17

Bibliografia **19**

A Topologia da Rede **21**

Prefácio

Universidade Federal de São Carlos (UFSCar) Sorocaba, 8 de março de 2016

Marcelo Frate
<frate@ifsp.edu.br>

Marcelo K. M. Marczuk
<marcelo.marczuk@dcomp.sor.ufscar.br>

Fábio L. Verdi
<verdi@ufscar.br>

Capítulo 1

Introdução

O OrchFlow, surge como uma ferramenta que tem como objetivo, funcionar como um *Middleware*, um software orquestrador para as SDN, com dois ou mais controladores baseadas no protocolo OpenFlow [2], proporcionando ao administrador da rede uma visão global, capaz de aprovisionar serviços, com monitoramento da topologia em tempo real. OrchFlow é capaz de receber solicitações de serviços através de uma interface Norte, *(Northbound)*, processá-las e então mapeá-las através de uma interface Sul *(Southbound)*, de forma a prover o serviço solicitado em uma rede controlada por múltiplos controladores OpenFlow. Assim, quando um determinado serviço é solicitado, o OrchFlow define como deverá ser o tratamento por parte dos diversos controladores na rede até que sejam aplicadas todas as regras OpenFlow aos elementos de rede, permitindo que os recursos estejam disponíveis.

1.1 Arquitetura do OrchFlow

Como se pode ver na Figura 1.1, o OrchFlow atuará como um agente integrador entre as diversas aplicações disponíveis na rede e os diferentes controladores OpenFlow, sob um mesmo controle administrativo, definido aqui como Domínio Administrativo.

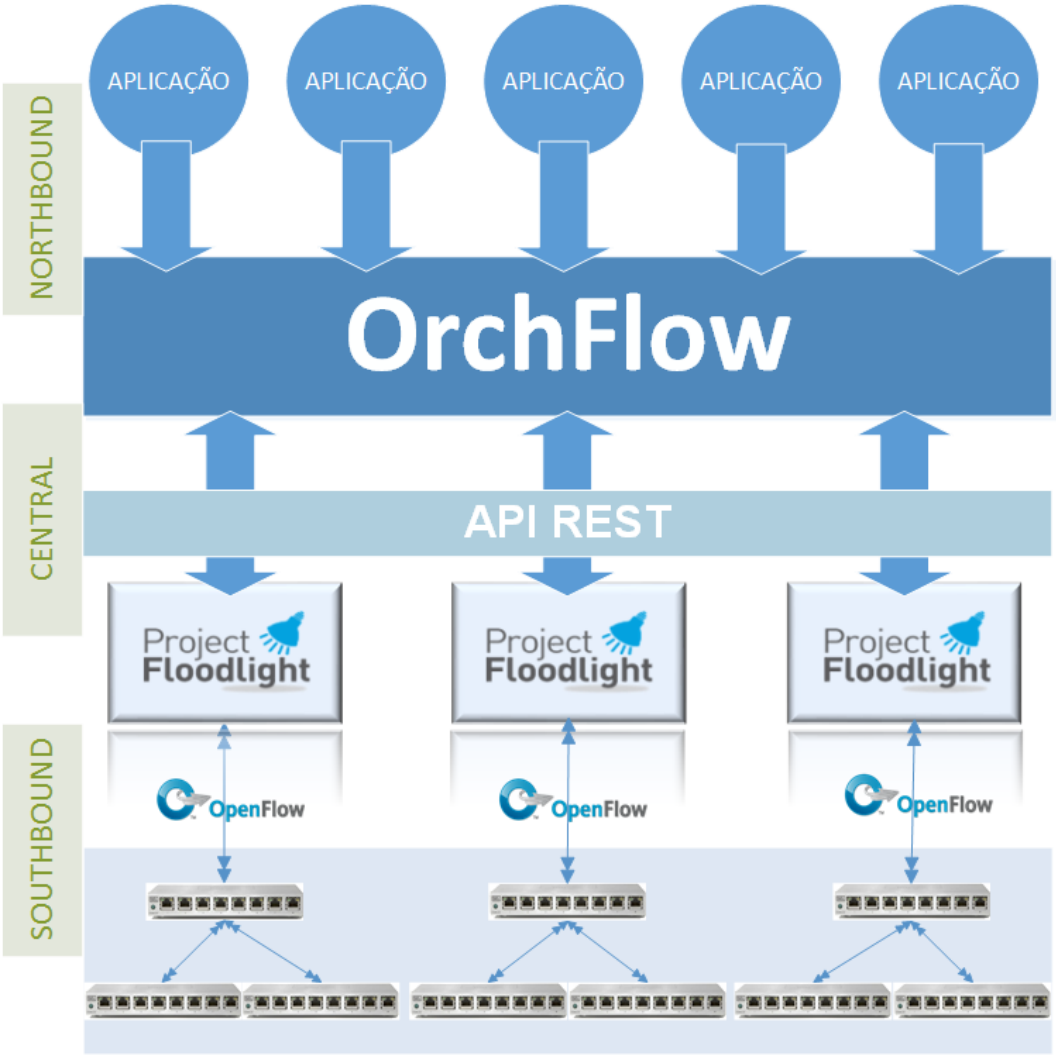

Figura 1.1: Arquitetura utilizada nos testes.

Capítulo 2

Funcionamento do OrchFlow

Na primeira camada da arquitetura estão os switches, formando toda a infraestrutura necessária para a comunicação entre os hosts. Para efeito de testes e uso deste manual, optamos por uma rede emulada através do emulador Mininet [1];

2.1 Detalhes da implementação

A arquitetura de rede proposta, tem para efeito de testes, um domínio administrativo composto por três subdomínios, com um controlador OpenFlow cada. Cada controlador é responsável pelo controle e gerenciamento de um único subdomínio e cabe ao OrchFlow a orquestração, gerencia e a visão completa do Domínio Administrativo.

Utilizamos aqui quatro máquinas virtuais (VM):

VM2, VM3 e VM4: Em cada uma, um controlador Floodlight já está instalado, configurado, funcionando e pronto para receber as chamadas dos switches das redes instaladas na VM 1;

VM1: Nesta VM, o usuário deverá emular a sua rede. Para facilitar o acesso e uso, deixamos um script pronto de uma rede emulada, contendo três subdomínios interligados em forma de anel, com 7 switches cada em forma de árvore e 4 hosts;

A Figura 2.1, ilustra a topologia utilizada nos testes e no desenvolvimento do OrchFlow.

Toda a infraestrutura necessária para a implantação e testes do OrchFlow está baseada na plataforma Linux. Assim, o LERIS, disponibiliza um servidor para acesso remoto com todas as máquinas virtuais, com o banco de dados Neo4j e com o OrchFlow já instalados e configurados, para que todos os testes sejam realizados.

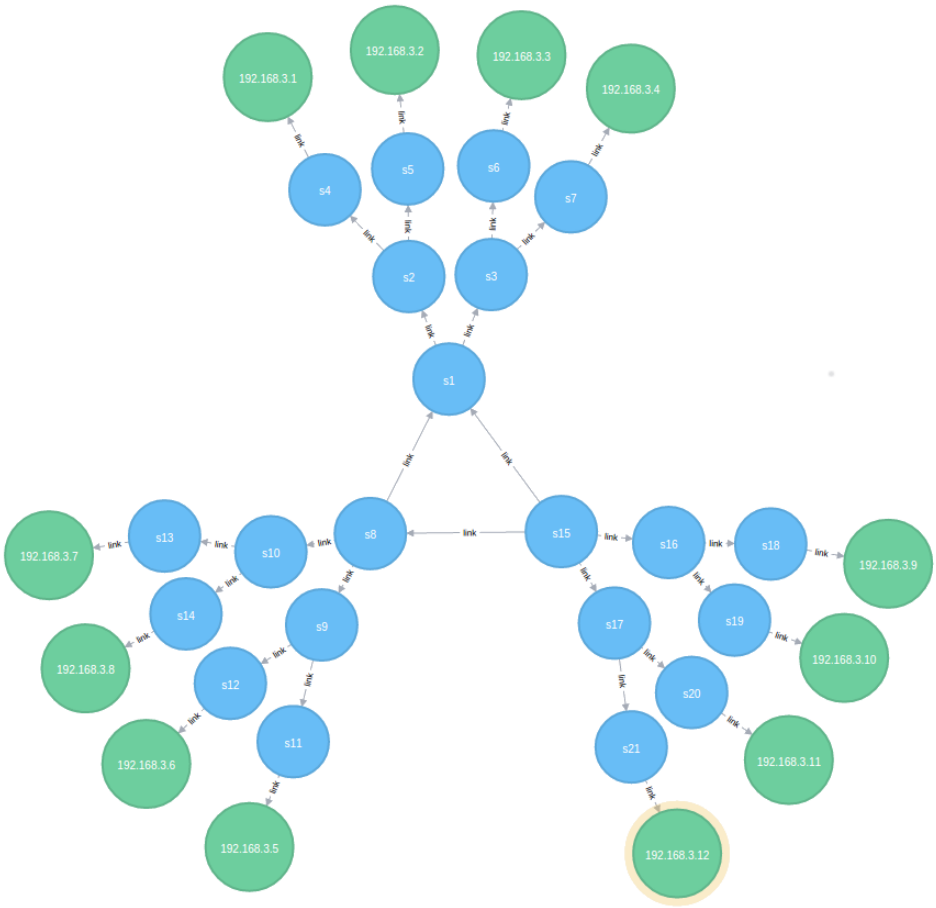

Figura 2.1: Topologia de rede.

Capítulo 3

Requisitos de Software

3.1 Funcionalidades

- Centralização da gestão da rede de computadores;
- Visão global do domínio administrativo;
- Criação de rotas estáticas;
- Criação de rotas dinâmicas em tempo real;
- Administração das rotas criadas;
- Cadastro e manutenção dos controladores da rede;

3.2 Requisitos

- javac 1.8.0_72;
- Banco de Dados Neo4j V2.3;
- Mininet V 2.2.1;
- OpenFlow V 1.3;
- ovs-vsctl (Open vSwitch) 2.0.2;
- 3 Controladores Floodlight;
- Módulo ARPReply instalado nos 3 controladores;
- Módulo Reactive instalado nos 3 controladores;

Capítulo 4

Inicialização do sistema de testes

Para executar o experimento será necessário o acesso, via SSH, ao servidor do LERIS através do endereço IP: 200.133.238.125, usuário e senha "orchflow". Como se pode ver na Figura 4.1 este servidor contém 4 máquinas virtuais, criadas especialmente para o salão de ferramentas do SBRC. Todas as máquinas estão a disposição do usuário, que poderá acessá-las via SSH com usuário e senha "mininet".

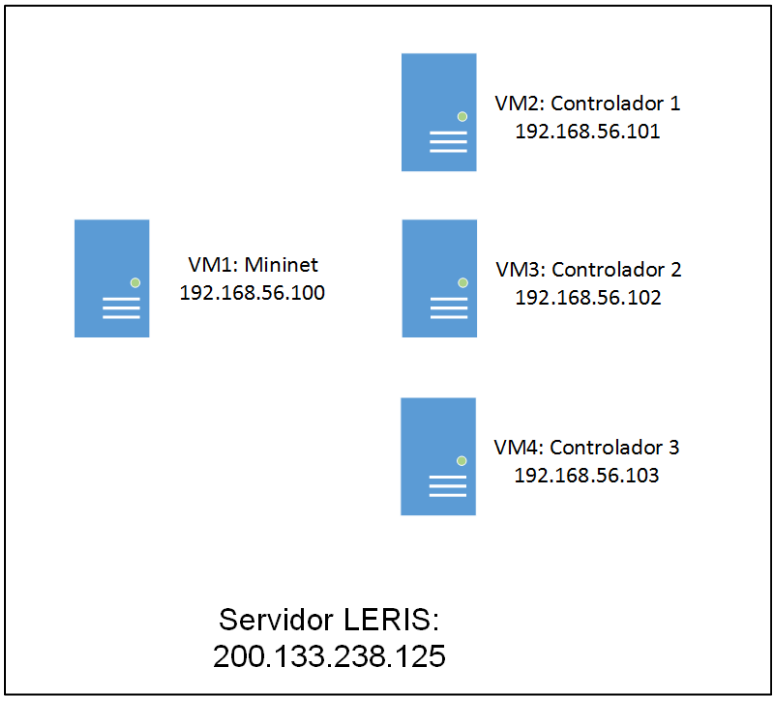

Figura 4.1: Ambiente LERIS.

Entretanto, para maior facilidade, deixamos executando os três controladores nas suas respectivas VMs(2, 3 e 4). Ficando portanto ao usuário, a necessidade de carregar a sua topologia na VM1, onde o emulador Mininet já está instalado.

4.1 Acesso ao Servidor LERIS

Para a realização dos testes, recomendamos usar um terminal de console linux, porém, o acesso poderá ser feito através de programas como o PUTTY em outros sistemas operacionais.

Para acessar o servidor LERIS abra um terminal e execute os seguintes comandos:

Acesso ao servidor LERIS

```
ssh -X orchflow@200.133.238.125
orchflow@200.133.238.125's password: orchflow
orchflow@ServerLeris:~$
```

Utilizamos o comando ssh -X, pois permite ao usuário orchflow rodar programas com interfaces gráficas, como por exemplo o navegador firefox. Estando com acesso ao servidor, você poderá acessar a 1ª máquina virtual (VM1), onde está instalado o Mininet e emular a sua própria rede ou executar um script preparado especificamente para este evento.

VM1: Mininet

```
orchflow@ServerLeris:~$ ssh -X mininet@192.168.56.101
mininet@192.168.56.101's password: mininet
Pmininet@mn1:~$ cd mininet/
mininet@mn1:~/mininet$ sudo ./custom/topologia.py
[sudo] password for mininet: mininet
```

Como foi dito anteriormente, o script topologia.py foi criado especificamente para este evento, caso o usuário queira criar a sua própria topologia, este terá a liberdade de fazê-la porém deve-se observar que as quatro VMs se comunicam através da rede interna 10.0.0.0.

O script utilizado para criar a topologia segue no Anexo A.

Espere até que seja executado todo o script, ele criará 12 (doze) hosts, 21 (vinte e um) switches, 3 (três) conexões com os controladores e 30 (trinta) links internos, entre os switches e hosts, além dos 3 (três) links externos que fazem a ligação entre os subdomínios.

Por fim, o script fará com que cada host tente executar um ping simples para um host qualquer, a fim de permitir que os controladores identifiquem cada um dos hosts na rede. Esse procedimento é necessário para demonstrarmos o correto funcionamento do OrchFlow, pois desabilitamos o modo *forward* dos controladores e é este modo que faz o reconhecimento automático de cada host.

Quando aparecer:"mininet> "significa que o script está completo e você poderá dar início aos testes propriamente dito.

Capítulo 5

Acessando o sistema OrchFlow

Agora abra o seu navegador e digite o seguinte endereço:
`http://200.133.238.125:8080/OrchFlow/`

Pronto, você estará na página inicial do OrchFlow, Figura 5.1, basta apertar o botão iniciar.

Vale lembrar, que o OrchFlow é uma aplicação WEB, instalada no servidor LERIS utilizando o Apache Tomcat, que poderá ser reiniciada em caso de problemas técnicos, no endereço:
`http://200.133.238.125:8080/manager/html`

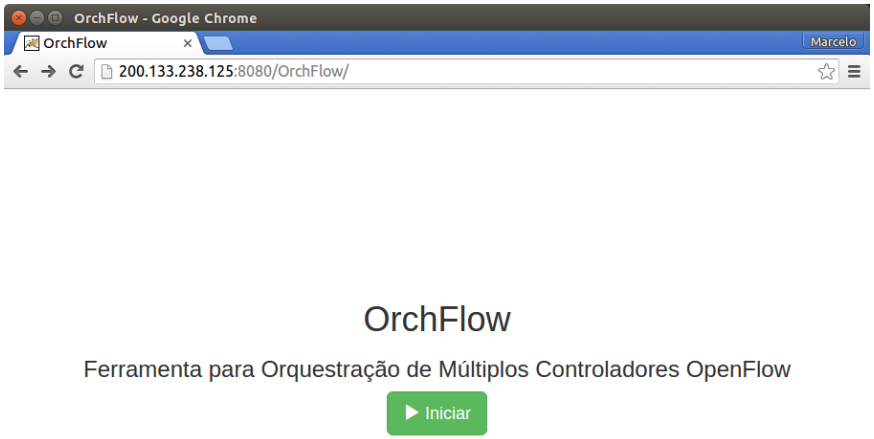

Figura 5.1: Página Inicial.

Na página seguinte, Figura 5.2, você deverá cadastrar os controladores utilizados na sua rede. Para o nosso experimento, os endereços utilizados são:

Controlador 1: 192.168.56.101 na porta 8085
Controlador 2: 192.168.56.102 na porta 8085
Controlador 3: 192.168.56.103 na porta 8085

Figura 5.2: Configurar.

Note que estes, são os endereços das máquinas virtuais criadas específicamente para este evento. Ao terminar o cadastro dos controladores clique em Concluir.

O OrchFlow, fará a busca pelos 3 controladores cadastrados e solicitará através da interface REST de cada um dos controladores os dados referentes a topologia de cada subdomínio, cadastrando-os no banco de dados Neo4j, previamente instalado e acessível pelo endereço:
http://200.133.238.125:7474/browser/

Note porém, que não há a necessidade de acessar o banco de dados de forma externa ao OrchFlow, pois como se pode ver na Figura 5.3, ele já dispõe de uma interface Neo4j integrada, do lado direito da página, onde é possível ver a topologia completa do domínio administrativo e executar os comandos desse banco de dados, tais como uma mudança de cor para os nós e arestas, mudança no tamanho dos nós, identificadores, entre outros. É possível realizar também comandos de busca de melhor caminho entre dois hosts, enfim, todos os comandos do Neo4j estão livres para uso e você pode ver como em:

http://neo4j.com/
http://neo4j.com/docs/stable/

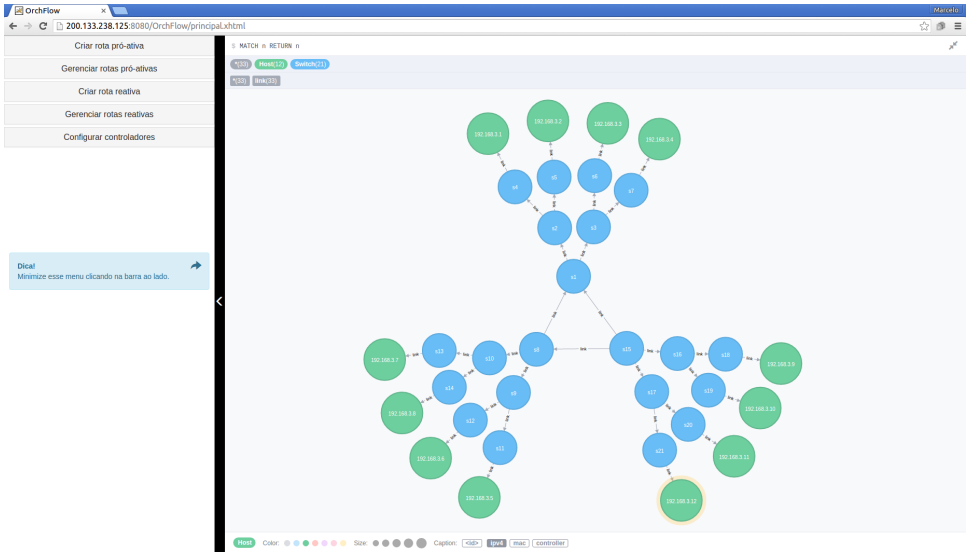

Figura 5.3: Principal.

Do lado esquerdo da interface é possível identificar os dois modos de operação do OrchFlow, protivo e reativo, no menu de opções e criar as regras conforme a necessidade da aplicação. Para ambos os modos o procedimento é o mesmo, basta preencher os campos necessários para a definição da rota.

Além disso é possível observar no menu as opções de gerência das rotas para os dois modos. Cada um deles afetará de forma diferente os switches, pois as rotas proativas são gravadas diretamente nas tabelas de fluxos dos switches, enquanto as rotas reativas são tabelas gravadas nos controladores que responderão aos switches sempre que novas conexões chegarem.

Por fim, é possível reconfigurar os controladores caso tenha digitado errado na tela anterior.

Capítulo 6

Utilizando o OrchFlow

Para a realização dos testes, alguns dos serviços de rede mais conhecidos estão sendo propostos como forma de validar esta ferramenta, tais como os serviços de roteamento fim a fim.

Pode-se iniciar os testes através do modo **Proativo:** Onde toda a programação é executada pelo OrchFlow e transferida aos controladores, via interface REST padrão todas as regras de fluxos, diretamente aos switches envolvidos.

Depois, pode-se realizar os mesmos testes no modo **Reativo:** Que ocorre quando um novo evento na rede, chegando através da interface Sul, faz com que, um controlador busque em sua programação a correta correspondência e faz a inserção dos fluxos necessários. Neste modo, os controladores OpenFlow, respondem diretamente às requisições de seu subdomínio e encaminham para a porta de saída correspondente ao subdomínio vizinho todas as requisições que não pertençam a este subdomínio, possibilitando assim a entrega de pacotes entre subdomínios, independentemente de quantos forem os subdomínios que compõem o domínio administrativo.

Em ambos os casos, ao clicar no menu Criar rota proativa ou reativa, você terá os campos OpenFlow para determinar as características de cada um dos fluxos, Figuras 6.1 e 6.2, Aqui você deverá escolher o host de origem e o host de destino, criar um nome para a rota e definir os parâmetros seguintes conforme a necessidade.

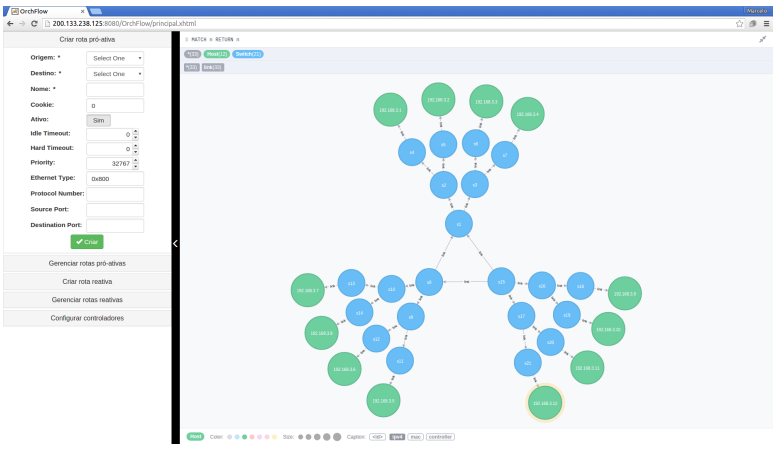

Figura 6.1: Proativo.

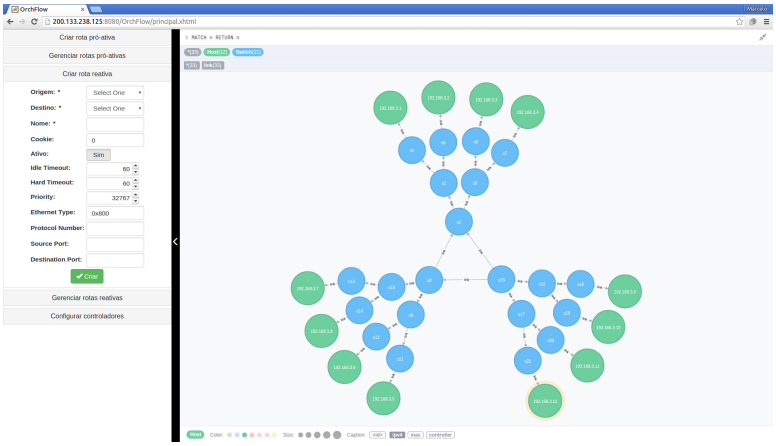

Figura 6.2: Reativo.

6.1 Ping

Para efeito de teste de conectividade, é possível criar uma rota com regras específicas. Vamos pingar do host 9 para o host 2, para isso vamos configurar o OrchFlow no modo proativo com os seguintes parâmetros:

Origem: 192.168.3.9
Destino: 192.168.3.2
Nome: PING-9-2
Cookie: 0
Ativo: Sim
Idle Timeout: 0
Hard Timeout: 0
Priority: 32767
Ethernet Type: 0x800
Protocol Number: 0x01
Porta Origem:
Porta Destino:

Aqui dissemos à rede que permita o tráfego ICMP, Protocol Number: 0x01, entre os dois hosts escolhidos, em duas redes diferentes, assim, o ping funcionará de um subdomínio para o outro, independentemente de origem e destino. Para verificar se a programação foi aplicada corretamente, volte ao terminal onde está sendo executado o mininet e digite o seguinte comando:
h9 ping h2
É possível executar também:
h2 ping h9
Percebe-se aqui que o OrchFlow configurou todos os switches, criando em suas tabelas de fluxos os caminhos de ida e volta, é possível verificar essa configuração diretamente nos controladores, através de suas interfaces, para isso, abra um novo terminal, acesse novamente o servidor LERIS e

abra o navegador firefox do servidor. Você terá acesso aos endereços dos controladores e poderá ver todas as configurações criadas para cada subdomínio.

Acesso ao servidor LERIS

```
ssh -X orchflow@200.133.238.125
orchflow@200.133.238.125's password: orchflow
orchflow@ServerLeris:~$ firefox &
```

Irá abrir o navegador conforme a Figura 6.3 com 3 abas já direcionadas para os 3 controladores, onde você terá acesso a cada subdomínio, pode-se observar também que o switch número 5, está com a rota de ida e volta criada para os hosts 9 e 2.

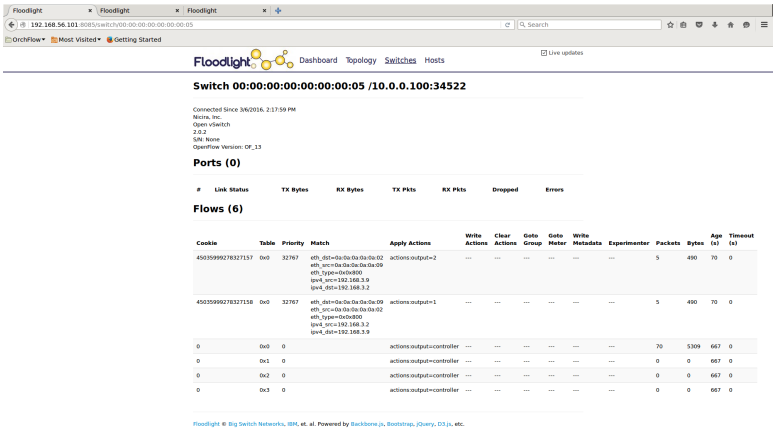

Figura 6.3: Controlador 1 - SW05

6.2 SSH

Aqui, iremos configurar o OrchFlow no modo reativo com os seguintes parâmetros:

Origem: 192.168.3.8
Destino: 192.168.3.1
Nome: SSH1
Cookie: 0
Ativo: Sim
Idle Timeout: 60
Hard Timeout: 60
Priority: 32767
Ethernet Type: 0x800
Protocol Number: 0x6
Porta Origem:
Porta Destino: 22

Repare que a porta origem ficará vazia, pois poderá vir de qualquer porta o pedido de conexão com o servidor.

Uma vez que tenha sido executado o comando os controladores receberão as programações devidas, porém, as regras ainda não estarão configuradas nos switches.

Acesse os hosts proceda com as seguintes configurações e comandos:

No host 1 que funcionará como um servidor, é preciso habilitar o serviço, verificar em qual porta o serviço será executado e permitir que o usuário root tenha acesso, para isso, edit o arquivo: `/etc/ssh/sshd_config`

Verifique as seguintes linhas, se não existirem, crie-as:

port 22
`#AllowUsers`
PermitRootLogin yes

Salve o arquivo e reinicie o serviço.

```
# /etc/init.d/ssh stop
# /etc/init.d/ssh start
```

no host 8, que funcionará aqui como o cliente, basta executar o comando:

```
#ssh root@192.168.3.1
```

a senha de acesso ao host é mininet.

Um detalhe interessante é que nenhum outro serviço estará habilitado para esta rota, nem mesmo o ping.

6.3 Servidor HTTP

Configure o OrchFlow no modo reativo com os seguintes parâmetros:

Origem: 192.168.3.10
Destino: 192.168.3.3
Nome: HTTP3
Cookie: 0
Ativo: Sim
Idle Timeout: 60
Hard Timeout: 60
Priority: 32767
Ethernet Type: 0x800
Protocol Number: 0x6
Porta Origem:
Porta Destino: 80

Repare que a porta origem ficará vazia, pois poderá vir de qualquer porta o pedido de conexão com o servidor.

Uma vez que tenha sido executado o comando os controladores receberão as programações devidas, porém, as regras ainda não estarão configuradas nos switches.

Acesse os hosts proceda com as seguintes configurações e comandos:

No host 3 que funcionará como um servidor, é preciso habilitar o serviço executando o seguinte comando:

```
#python -m SimpleHTTPServer 80 &
```

no host 8, que funcionará aqui como o cliente, basta executar o comando:

```
#wget -O - 192.168.3.3
```

se tudo correu bem, você receberá uma confirmação:

Connecting to 192.168.3.3:80... connected.
HTTP request sent, awaiting response... 200 OK

Um detalhe interessante é que nenhum outro serviço estará habilitado para esta rota, nem mesmo o ping.

Caso queira finalizar o serviço no servidor, utilize o seguinte comando:

```
#kill %python
```

6.4 Servidor FTP

Configure o OrchFlow no modo reativo com os seguintes parâmetros:

Origem: 192.168.3.11
Destino: 192.168.3.4
Nome: FTP4
Cookie: 0
Ativo: Sim
Idle Timeout: 60
Hard Timeout: 60
Priority: 32767
Ethernet Type: 0x800
Protocol Number: 0x6
Porta Origem:
Porta Destino: 21

Repare que a porta origem ficará vazia, pois poderá vir de qualquer porta o pedido de conexão com o servidor.

Uma vez que tenha sido executado o comando os controladores receberão as programações devidas, porém, as regras ainda não estarão configuradas nos switches.

Acesse os hosts proceda com as seguintes configurações e comandos:

No host 4 que funcionará como um servidor, é preciso habilitar o serviço.

```
# inetd &
```

no host 11, que funcionará aqui como o cliente, basta executar o comando:

```
#ftp 192.168.3.4
```

o usuário e a senha de acesso ao host é mininet.

Um detalhe interessante é que nenhum outro serviço estará habilitado para esta rota, nem mesmo o ping.

Bibliografia

[1] Bob Lantz, Brandon Heller e N McKeown. "A network in a laptop: rapid prototyping for software-defined networks". Em: *... Workshop on Hot Topics in Networks* (2010), pp. 1–6. ISSN: 1450304095. DOI: 10.1145/1868447.1868466. URL: http://dl.acm.org/citation.cfm?id=1868466.

[2] Nick McKeown et al. "OpenFlow". Em: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74. ISSN: 01464833. DOI: 10.1145/1355734.1355746.

Apêndice A

Topologia da Rede

```
#!/usr/bin/python

import re
import sys

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, UserSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info, error
from mininet.link import Link, TCLink, Intf
from mininet.util import quietRun

def topology():
"Create a network."
net = Mininet( controller=RemoteController, link=TCLink, switch=OVSKernelSwitch )

print "*** Creating nodes"
h1 = net.addHost( 'h1', mac='0a:0a:0a:0a:0a:01', ip='192.168.3.1/24' )
h2 = net.addHost( 'h2', mac='0a:0a:0a:0a:0a:02', ip='192.168.3.2/24' )
h3 = net.addHost( 'h3', mac='0a:0a:0a:0a:0a:03', ip='192.168.3.3/24' )
h4 = net.addHost( 'h4', mac='0a:0a:0a:0a:0a:04', ip='192.168.3.4/24' )

h5 = net.addHost( 'h5', mac='0a:0a:0a:0a:0a:05', ip='192.168.3.5/24' )
h6 = net.addHost( 'h6', mac='0a:0a:0a:0a:0a:06', ip='192.168.3.6/24' )
h7 = net.addHost( 'h7', mac='0a:0a:0a:0a:0a:07', ip='192.168.3.7/24' )
h8 = net.addHost( 'h8', mac='0a:0a:0a:0a:0a:08', ip='192.168.3.8/24' )

h9 = net.addHost( 'h9', mac='0a:0a:0a:0a:0a:09', ip='192.168.3.9/24' )
h10 = net.addHost( 'h10', mac='0a:0a:0a:0a:0a:0a', ip='192.168.3.10/24' )
h11 = net.addHost( 'h11', mac='0a:0a:0a:0a:0a:0b', ip='192.168.3.11/24' )
h12 = net.addHost( 'h12', mac='0a:0a:0a:0a:0a:0c', ip='192.168.3.12/24' )
```

```
s1 = net.addSwitch( 's1', protocols='OpenFlow13', listenPort=6671, mac='00:00:00:00:00:01' )
s2 = net.addSwitch( 's2', protocols='OpenFlow13', listenPort=6672, mac='00:00:00:00:00:02' )
s3 = net.addSwitch( 's3', protocols='OpenFlow13', listenPort=6673, mac='00:00:00:00:00:03' )
s4 = net.addSwitch( 's4', protocols='OpenFlow13', listenPort=6674, mac='00:00:00:00:00:04' )
s5 = net.addSwitch( 's5', protocols='OpenFlow13', listenPort=6675, mac='00:00:00:00:00:05' )
s6 = net.addSwitch( 's6', protocols='OpenFlow13', listenPort=6676, mac='00:00:00:00:00:06' )
s7 = net.addSwitch( 's7', protocols='OpenFlow13', listenPort=6677, mac='00:00:00:00:00:07' )

s8 = net.addSwitch( 's8', protocols='OpenFlow13', listenPort=6678, mac='00:00:00:00:00:08' )
s9 = net.addSwitch( 's9', protocols='OpenFlow13', listenPort=6679, mac='00:00:00:00:00:09' )
s10 = net.addSwitch( 's10', protocols='OpenFlow13', listenPort=6680, mac='00:00:00:00:00:0a' )
s11 = net.addSwitch( 's11', protocols='OpenFlow13', listenPort=6681, mac='00:00:00:00:00:0b' )
s12 = net.addSwitch( 's12', protocols='OpenFlow13', listenPort=6682, mac='00:00:00:00:00:0c' )
s13 = net.addSwitch( 's13', protocols='OpenFlow13', listenPort=6683, mac='00:00:00:00:00:0d' )
s14 = net.addSwitch( 's14', protocols='OpenFlow13', listenPort=6684, mac='00:00:00:00:00:0e' )

s15 = net.addSwitch( 's15', protocols='OpenFlow13', listenPort=6685, mac='00:00:00:00:00:0f' )
s16 = net.addSwitch( 's16', protocols='OpenFlow13', listenPort=6686, mac='00:00:00:00:00:10' )
s17 = net.addSwitch( 's17', protocols='OpenFlow13', listenPort=6687, mac='00:00:00:00:00:11' )
s18 = net.addSwitch( 's18', protocols='OpenFlow13', listenPort=6688, mac='00:00:00:00:00:12' )
s19 = net.addSwitch( 's19', protocols='OpenFlow13', listenPort=6689, mac='00:00:00:00:00:13' )
s20 = net.addSwitch( 's20', protocols='OpenFlow13', listenPort=6690, mac='00:00:00:00:00:14' )
s21 = net.addSwitch( 's21', protocols='OpenFlow13', listenPort=6691, mac='00:00:00:00:00:15' )

c1 = net.addController( 'c1', controller=RemoteController, ip='10.0.0.101', port=6653 )
c2 = net.addController( 'c2', controller=RemoteController, ip='10.0.0.102', port=6653 )
c3 = net.addController( 'c3', controller=RemoteController, ip='10.0.0.103', port=6653 )

print "*** Creating links"
net.addLink(s1, s2, 1, 1)
net.addLink(s1, s3, 2, 1)
net.addLink(s2, s4, 2, 1)
net.addLink(s2, s5, 3, 1)
net.addLink(s3, s6, 2, 1)
net.addLink(s3, s7, 3, 1)

net.addLink(s8, s9, 1, 1)
net.addLink(s8, s10, 2, 1)
net.addLink(s9, s11, 2, 1)
net.addLink(s9, s12, 3, 1)
net.addLink(s10, s13, 2, 1)
net.addLink(s10, s14, 3, 1)

net.addLink(s15, s16, 1, 1)
net.addLink(s15, s17, 2, 1)
net.addLink(s16, s18, 2, 1)
net.addLink(s16, s19, 3, 1)
```

```
net.addLink(s17, s20, 2, 1)
net.addLink(s17, s21, 3, 1)

net.addLink(h1, s4, 0, 2)
net.addLink(h2, s5, 0, 2)
net.addLink(h3, s6, 0, 2)
net.addLink(h4, s7, 0, 2)
net.addLink(h5, s11, 0, 2)
net.addLink(h6, s12, 0, 2)
net.addLink(h7, s13, 0, 2)
net.addLink(h8, s14, 0, 2)
net.addLink(h9, s18, 0, 2)
net.addLink(h10, s19, 0, 2)
net.addLink(h11, s20, 0, 2)
net.addLink(h12, s21, 0, 2)

print "*** Starting network"
net.build()
c1.start()
c2.start()
c3.start()

s1.start( [c1] )
s2.start( [c1] )
s3.start( [c1] )
s4.start( [c1] )
s5.start( [c1] )
s6.start( [c1] )
s7.start( [c1] )

s8.start( [c2] )
s9.start( [c2] )
s10.start( [c2] )
s11.start( [c2] )
s12.start( [c2] )
s13.start( [c2] )
s14.start( [c2] )

s15.start( [c3] )
s16.start( [c3] )
s17.start( [c3] )
s18.start( [c3] )
s19.start( [c3] )
s20.start( [c3] )
s21.start( [c3] )

s1.cmd('ovs-vsctl add-port s1 s1-ext1 – set interface s1-ext1 type=patch options:peer=s8-ext1')
```

```
s1.cmd('ovs-vsctl add-port s1 s1-ext3 – set interface s1-ext3 type=patch options:peer=s15-ext3')
s8.cmd('ovs-vsctl add-port s8 s8-ext1 – set interface s8-ext1 type=patch options:peer=s1-ext1')
s8.cmd('ovs-vsctl add-port s8 s8-ext2 – set interface s8-ext2 type=patch options:peer=s15-ext2')
s15.cmd('ovs-vsctl add-port s15 s15-ext2 – set interface s15-ext2 type=patch options:peer=s8-ext2')
s15.cmd('ovs-vsctl add-port s15 s15-ext3 – set interface s15-ext3 type=patch options:peer=s1-ext3')

s1.cmdPrint('ovs-vsctl show')
h1.cmdPrint('ping 192.168.3.12 -c 1')
h2.cmdPrint('ping 192.168.3.12 -c 1')
h3.cmdPrint('ping 192.168.3.12 -c 1')
h4.cmdPrint('ping 192.168.3.12 -c 1')
h5.cmdPrint('ping 192.168.3.12 -c 1')
h6.cmdPrint('ping 192.168.3.12 -c 1')
h7.cmdPrint('ping 192.168.3.12 -c 1')
h8.cmdPrint('ping 192.168.3.12 -c 1')
h9.cmdPrint('ping 192.168.3.12 -c 1')
h10.cmdPrint('ping 192.168.3.12 -c 1')
h11.cmdPrint('ping 192.168.3.12 -c 1')
h12.cmdPrint('ping 192.168.3.1 -c 1')

print "*** Running CLI"
CLI( net )

print "*** Stopping network"
net.stop()

if __name__ == '__main__':
setLogLevel( 'info' )
topology()
```