

---

---

# OrchFlow

Uma Ferramenta para Orquestração de  
Múltiplos Controladores OpenFlow

---

---

**OrchFlow**  
Manual do usuário



LERIS - Laboratory of Studies in Networks,  
Innovation and Software  
UFSCar - Sorocaba  
<http://leris.sor.ufscar.br/>

**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  
Instituto Federal de São Paulo (IFSP) - Boituva

**Participante(s):**

Marcelo Frate  
Marcelo K. M. Marczuk

**Orientador(s):**

Fábio L. Verdi

**Cópias:** 1

**Numero de páginas:** 20

**Data de Publicação:**

04 de Abril 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

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Arquitetura do OrchFlow . . . . .	1
<b>2</b>	<b>Funcionamento do OrchFlow</b>	<b>2</b>
2.1	Detalhes da implementação . . . . .	2
<b>3</b>	<b>Requisitos de Software</b>	<b>4</b>
3.1	Funcionalidades . . . . .	4
3.2	Requisitos . . . . .	4
<b>4</b>	<b>Inicialização do sistema de testes</b>	<b>5</b>
4.1	Acesso ao Servidor LERIS . . . . .	6
<b>5</b>	<b>Acessando o sistema OrchFlow</b>	<b>7</b>
<b>6</b>	<b>Utilizando o OrchFlow</b>	<b>9</b>
6.1	Ping . . . . .	10
6.2	SSH . . . . .	11
6.3	Servidor HTTP . . . . .	14
6.4	Servidor FTP . . . . .	14
	<b>Bibliografia</b>	<b>16</b>
<b>A</b>	<b>Topologia da Rede</b>	<b>17</b>

# Capítulo 1

## Introdução

OrchFlow, uma ferramenta que tem como objetivo funcionar como um *Middleware*, um software orquestrador para as redes SDN baseadas no protocolo OpenFlow [2], 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. Quando um determinado serviço é solicitado, o OrchFlow define como deverá ser o tratamento por parte dos diversos controladores até que sejam aplicadas as regras OpenFlow a todos os elementos de rede.

### 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. Para fins de demonstração neste Salão, estamos utilizando apenas controladores Floodlight.

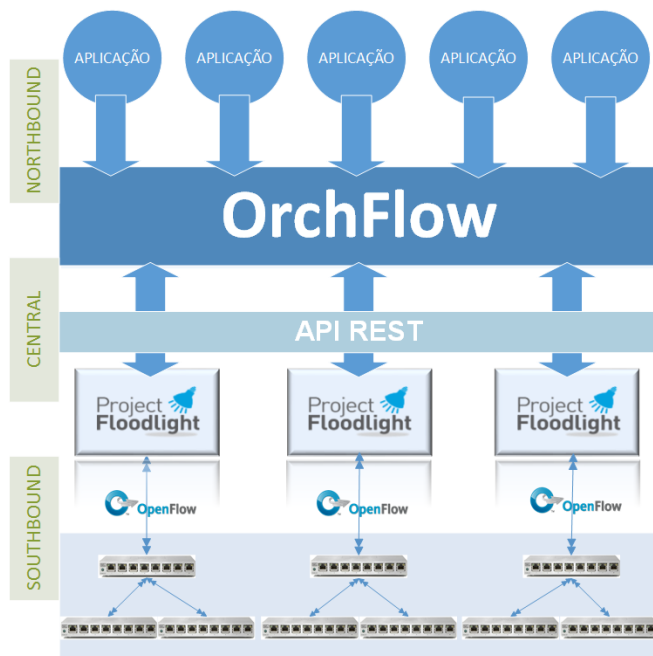


Figura 1.1: Arquitetura utilizada nos testes.

## Capítulo 2

# Funcionamento do OrchFlow

Como mostrado na Figura 1.1, a primeira camada da arquitetura contém 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 aqui proposta contém 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.

Utiliza-se 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 VM1;

**VM1:** Nesta VM, o usuário deverá emular a sua rede. Para facilitar o acesso e uso, foi preparado um script 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 o OrchFlow já instalados e configurados para que todos os testes sejam realizados.

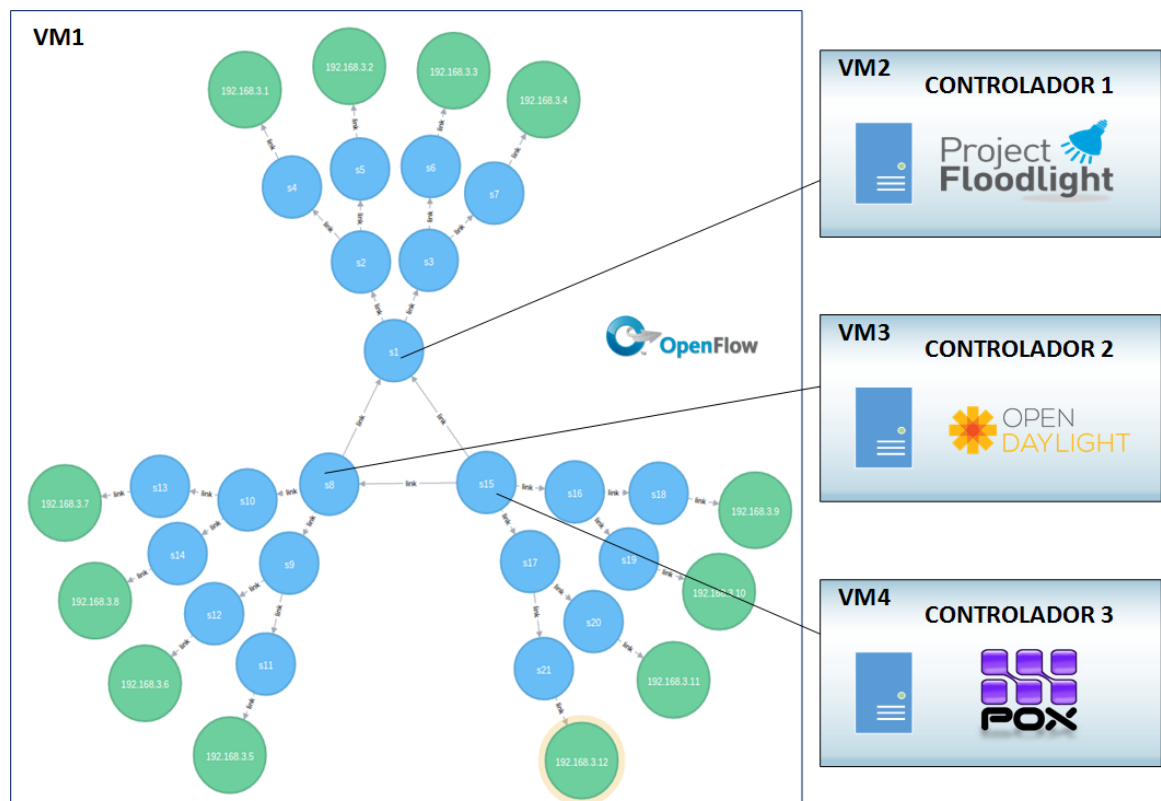


Figura 2.1: Topologia de rede.

## Capítulo 3

# Requisitos de Software

### 3.1 Funcionalidades

- Centralização da gestão da rede de computadores;
- Visão global do domínio administrativo;
- Criação de serviços através do modo proativo;
- Criação de serviços através do modo reativo em tempo real;
- Administração dos serviços cadastrados;
- Cadastro e manutenção dos controladores da rede;

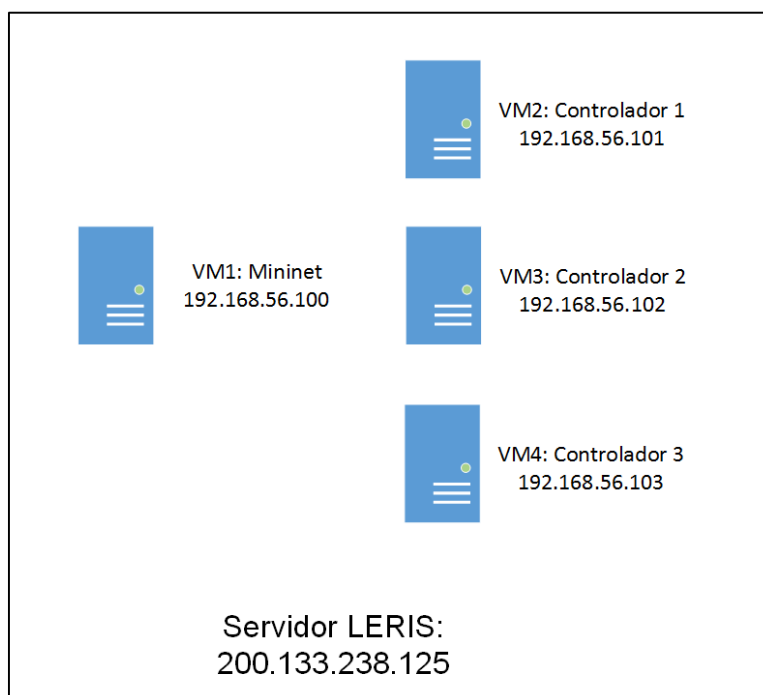
### 3.2 Requisitos

- 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 "orchflow" e senha "OrchFlow2016". Como se pode ver na Figura 4.1 este servidor contém 4 máquinas virtuais, criadas especialmente para o Salão de Ferramentas do SBRC 2016. Todas as máquinas estão a disposição do usuário, que poderá acessá-las via SSH com usuário e senha "mininet".



**Figura 4.1:** Ambiente LERIS.

Para maior facilidade, os três controladores Floodlight já estão sendo executados nas suas respectivas VMs (2, 3 e 4). Ficando portanto ao usuário, a necessidade de carregar apenas a sua topologia na VM1, onde o emulador Mininet já está instalado.

Importante observar que em caso de problemas de acesso ao servidor LERIS, ou erros de funcionamento do OrchFlow, o usuário poderá acessar um servidor de backup, disponibilizado pelo IFSP - Campus Boituva, no endereço IP: **200.133.218.83** com as mesmas configurações de acesso incluindo usuários e senhas.



## 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: OrchFlow2016
orchflow@ServerLeris:~$
```

Recomenda-se o comando `ssh -X`, pois permite ao usuário `orchflow` rodar programas com interfaces gráficas, como por exemplo o navegador `firefox` que será utilizado para acessar o *dashboard* dos controladores. 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@192.168.56.100
mininet@192.168.56.101's password: mininet
mininet@mininet:~$ sudo mn -c
[sudo] password for mininet: mininet
mininet@mininet:~$ sudo ./mininet/custom/topologia.py
```

O comando "`mn -c`" está sendo utilizado para limpar possíveis configurações deixadas pelo mininet e como 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 o modo *forward* dos controladores foi desabilitado e é este modo que faz o reconhecimento automático de cada host.

Quando aparecer o prompt do mininet ( `mininet>` ) significa que o script executou completamente, que a rede está completa e pronta para o início dos testes.

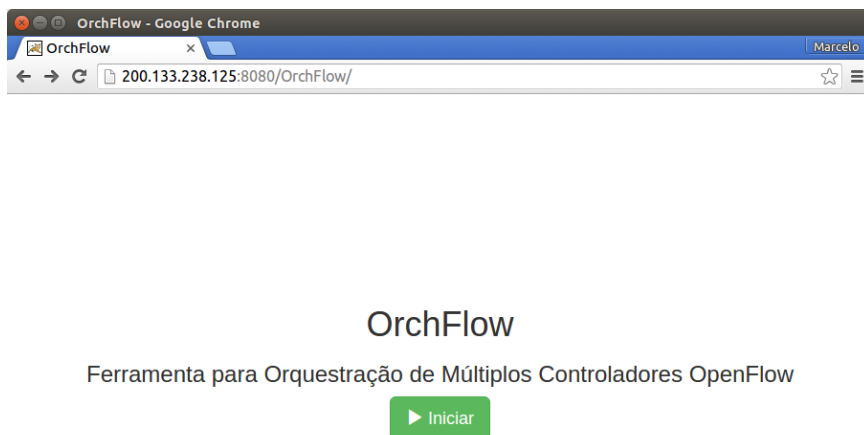
## Capítulo 5

# Acessando o sistema OrchFlow

Para acessar o OrchFlow, abra o seu navegador preferido e digite o seguinte endereço:

`http://200.133.238.125:8080/OrchFlow/`

A página inicial do OrchFlow abrirá conforme a Figura 5.1, clique no botão iniciar.



**Figura 5.1:** Página Inicial.

É importante lembrar que o OrchFlow é uma aplicação WEB, instalada no servidor LERIS utilizando o Apache Tomcat e que poderá ser reiniciada em caso de problemas técnicos.

Na página seguinte, como se pode ver na Figura 5.2, o usuário 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

**Cadastrar Controladores**

Adicione os controlador que serão orquestrados pelo OrchFlow

Nome:

IP: \*

Porta: \*

Nome	IP	Porta	Ação
C1	192.168.56.101	8085	<input type="button" value="Remover"/>
C2	192.168.56.102	8085	<input type="button" value="Remover"/>
C3	192.168.56.103	8085	<input type="button" value="Remover"/>

**Figura 5.2:** Configurar.

Note que estes são os endereços das máquinas virtuais criadas especificamente 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 nas Figuras 6.1 e 6.2, 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: mudança de cor para os nós e arestas, mudança no tamanho dos nós e identificadores. É possível realizar também comandos de busca de melhor caminho entre dois hosts. Em resumo, todos os comandos do Neo4j estão livres para uso e você pode ver como em:

<http://neo4j.com/>

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

Do lado esquerdo da interface é possível identificar os dois modos de operação do OrchFlow, proativo e reativo 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 do serviço fim a fim.

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

Por fim, é possível reconfigurar os controladores caso tenha ocorrido algum erro de digitação na tela anterior.

## Capítulo 6

# Utilizando o OrchFlow

Para a realização dos testes, um dos serviços de rede mais conhecidos está sendo proposto como forma de validar esta ferramenta: o serviço 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. Neste modo, todas as regras de fluxos são enviadas imediatamente aos switches envolvidos no serviço a ser estabelecido.

Após isso, pode-se realizar os mesmos testes no modo **Reativo** onde as regras são criadas e enviadas para os controladores, porém, diferentemente do modelo Proativo, elas não são implantadas imediatamente nos switches. Tais regras serão implantadas em resposta a um evento *packet-in*. Este evento é enviado através da interface Sul (OpenFlow) para o controlador do subdomínio. O controlador fará então uma busca em sua programação e, caso haja uma correspondência, ele adiciona as regras nas tabelas de fluxos de todos os switches de seu subdomínio, necessários para o estabelecimento do serviço solicitado.

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.

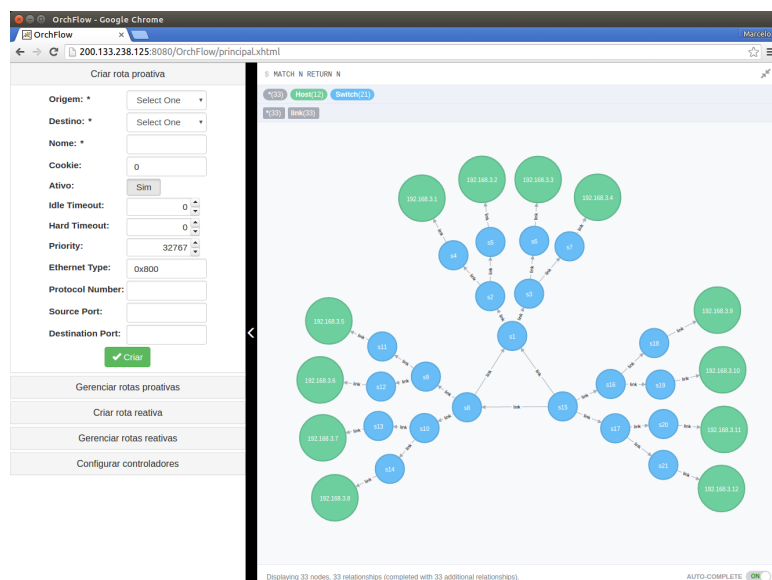


Figura 6.1: Proativo.

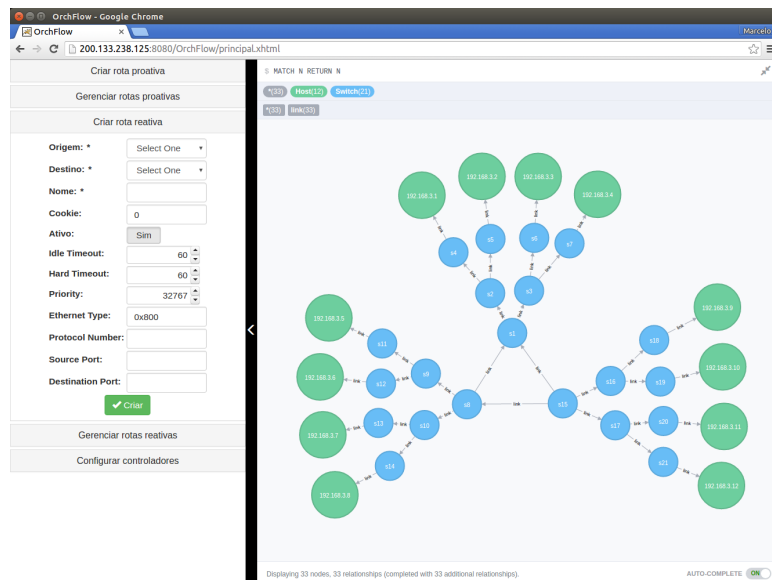


Figura 6.2: Reativo.

Deixamos alguns exemplos de caso de uso, que servirão como modelo para o usuário caso queira criar ou realizar os seus próprios experimentos.

## 6.1 Ping

Para efeito de teste de conectividade, é possível criar uma rota com regras específicas e demonstrar através do comando ping. É possível testar a conectividade entre dois hosts de subdomínios diferentes como por exemplo entre o host 9 e o host 2, nos dois modos de operação, proativo e reativo.

Primeiro o usuário precisa escolher o modo (aqui escolhemos o modo proativo, Fig. 6.1) e depois configurar o OrchFlow 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: 20  
Priority: 32767  
Ethernet Type: 0x800  
Protocol Number: 0x01  
Porta Origem:  
Porta Destino:

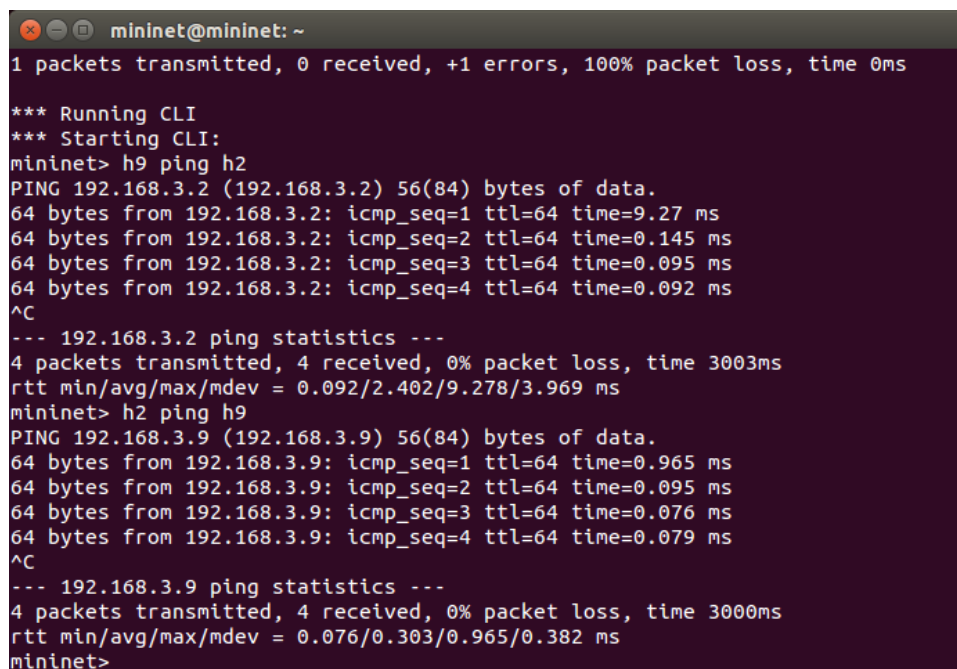
Tal configuração permitirá o tráfego ICMP, (*Protocol Number: 0x01*), entre os dois hosts escolhidos, hosts estes de duas redes diferentes. Assim, o ping funcionará de um subdomínio para outro, independentemente de origem e destino. Para verificar se a programação foi aplicada corretamente o usuário poderá voltar ao terminal onde está sendo executado o mininet e digitar o seguinte comando:

h9 ping h2

É possível executar também:

h2 ping h9

Espera-se obter um resultado parecido com o da figura 6.3



```
mininet@mininet: ~
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

*** Running CLI
*** Starting CLI:
mininet> h9 ping h2
PING 192.168.3.2 (192.168.3.2) 56(84) bytes of data.
64 bytes from 192.168.3.2: icmp_seq=1 ttl=64 time=9.27 ms
64 bytes from 192.168.3.2: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 192.168.3.2: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 192.168.3.2: icmp_seq=4 ttl=64 time=0.092 ms
^C
--- 192.168.3.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.092/2.402/9.278/3.969 ms
mininet> h2 ping h9
PING 192.168.3.9 (192.168.3.9) 56(84) bytes of data.
64 bytes from 192.168.3.9: icmp_seq=1 ttl=64 time=0.965 ms
64 bytes from 192.168.3.9: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 192.168.3.9: icmp_seq=3 ttl=64 time=0.076 ms
64 bytes from 192.168.3.9: icmp_seq=4 ttl=64 time=0.079 ms
^C
--- 192.168.3.9 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.076/0.303/0.965/0.382 ms
mininet>
```

Figura 6.3: Ping entre h9 e h2.

Percebe-se aqui que o OrchFlow configurou através dos controladores todos os seis switches (s5, s2, s1, s15, s16 e s18) que compõem o caminho entre os hosts h9 e h2, 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 dashboard 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 &
```

Tal comando fará com que abra o navegador firefox do servidor LERIS conforme a Figura 6.4 com 3 abas já direcionadas para os 3 controladores, onde você terá acesso a cada subdomínio e poderá obter as informações sobre as regras em cada um dos switches desse subdomínio. Pode-se observar também especificamente no switch número 5 que as regras de ida e volta foram criadas para os hosts 9 e 2, bem como em todos os outros switches envolvidos no serviço de estabelecimento da rota fim a fim.

## 6.2 SSH

Para este serviço, iremos configurar o OrchFlow no modo reativo com os seguintes parâmetros:

Origem: 192.168.3.8

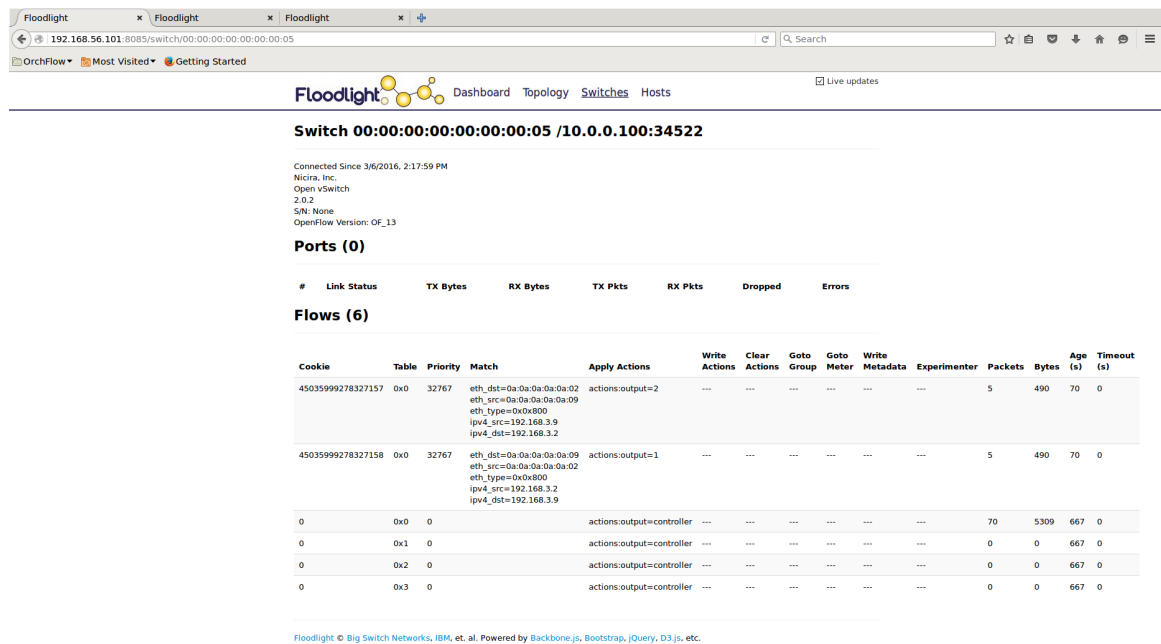


Figura 6.4: Controlador 1 - SW05

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. Pode-se verificar diretamente nos switches (s4, s2, s1, s8, s10, s14).

Volte ao terminal do mininet para acessar os hosts e proceda com as seguintes etapas de configurações e comandos:

Para o host 1 que funcionará como um servidor, foi preciso habilitar o serviço, verificando em qual porta o serviço seria executado e possibilitando que o usuário root obtivesse acesso, para isso, foi editado o arquivo:

```
/etc/ssh/sshd_config
```

Foi observada a existência das seguintes linhas:

```
port 22
```

```
#AllowUsers
```

PermitRootLogin yes

O arquivo foi salvo e reiniciado o serviço SSH no host, através dos comandos.

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

Note, que o usuário não precisará mais executar os procedimentos acima, pois neste experimento os hosts já estão preparados.

Entretanto, para o host 8 que funcionará como um cliente, o usuário precisará acessá-lo e prosseguir com o acesso ao host 1, servidor, através dos seguintes comandos:

No terminal do Mininet, utilize o comando:

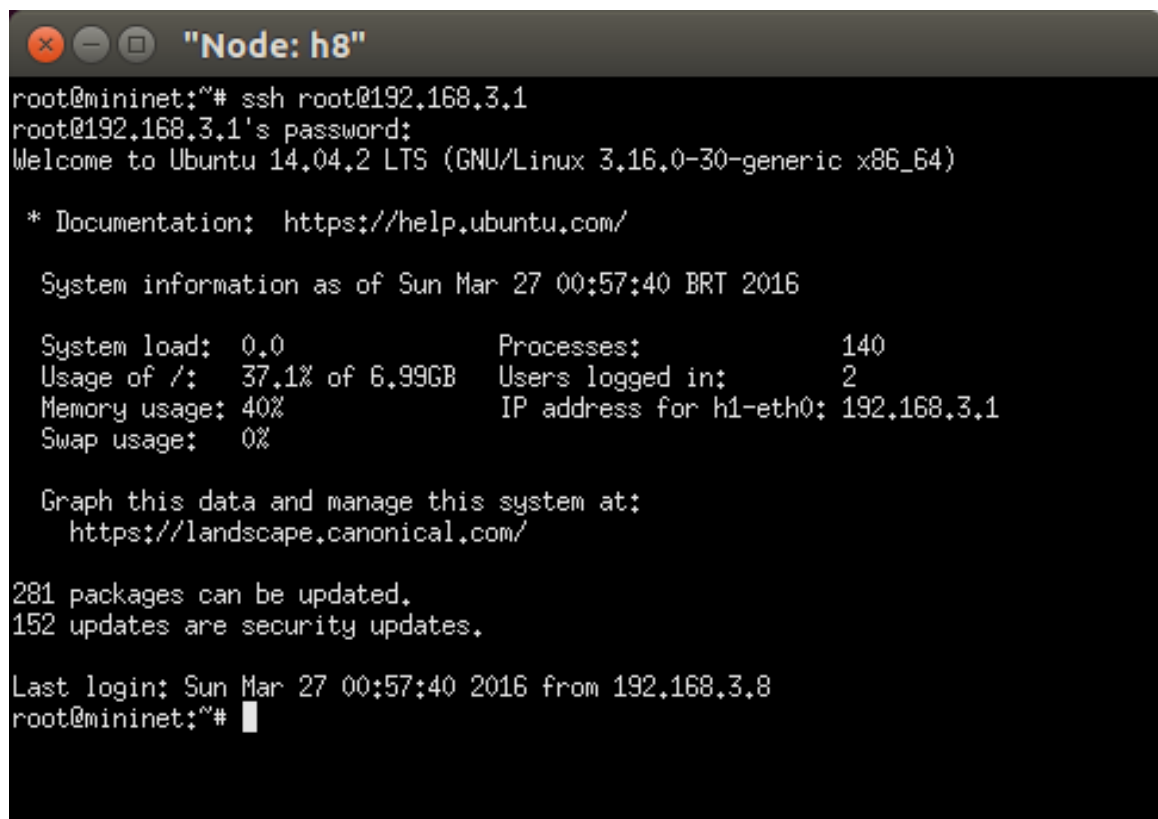
```
xterm h8
```

Abrirá uma nova janela de terminal conforme a Figura 6.5, para o host 8 que terá acesso ao host 1 através do 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.



```
"Node: h8"  
root@mininet:~# ssh root@192.168.3.1  
root@192.168.3.1's password:  
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.16.0-30-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com/  
  
System information as of Sun Mar 27 00:57:40 BRT 2016  
  
System load:  0.0                Processes:            140  
Usage of /:   37.1% of 6.99GB    Users logged in:     2  
Memory usage: 40%              IP address for h1-eth0: 192.168.3.1  
Swap usage:   0%  
  
Graph this data and manage this system at:  
https://landscape.canonical.com/  
  
281 packages can be updated.  
152 updates are security updates.  
  
Last login: Sun Mar 27 00:57:40 2016 from 192.168.3.8  
root@mininet:~#
```

Figura 6.5: Serviço SSH



## 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 correr bem, o usuário 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 proativo 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, encaminhando as regras diretamente aos switches. Percebe-se aqui que o OrchFlow configurou através dos controladores todos os seis switches (s20, s17, s15, s1, s3 e s7) que compõem o caminho entre os hosts h11 e h4, criando em suas tabelas de fluxos os caminhos de ida e volta. É possível verificar essa configuração diretamente nos controladores, através de suas interfaces.

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 como um 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()

```