Criação de um *learning switch L3* usando a plataforma Kytos

Visão Geral

Neste tutorial você irá criar um experimento SDN usando o testbed FIBRE e o controlador *OpenFlow* Kytos, para se familiarizar com o orquestrador de experimentos OCF e com a CLI do controlador Kytos.

O que você irá aprender

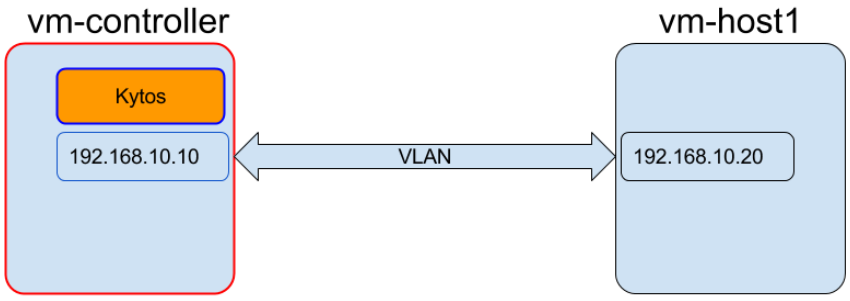
1. Criar máquinas virtuais usando o orquestrador OCF
2. Alocar uma topologia virtual e iniciar um experimento
3. Gerenciar um controlador *OpenFlow*
4. Visualizar fluxos e estatísticas *OpenFlow*

Pré-requisitos

1. Ter uma conta criada no FIBRE (por exemplo em: http://portal.rnp.fibre.org.br)
2. Estar conectado à VPN do FIBRE (https://fibre.org.br/start-using-fibre/getting-access/)
3. Ter conhecimentos básicos de experimentação (Criar projeto, criar slice: https://fibre.org.br/start-using-fibre/your-first-experiment/ocf-hello-world/)
4. Cliente SSH (em ambientes Windows, sugerimos o uso da *suit* Moba Xterm ou PuTTY)

Topologia

A topologia proposta para este tutorial forma uma rede *layer 2*, com dois *hosts*. Um dos *hosts* desempenha o papel de controlador da rede SDN, enquanto o outro *host* será um par para podermos observar o comportamento da rede. O objetivo deste experimento é utilizar a plataforma Kytos para operar o controlador SDN e instalar regras *openflow* que atuam sobre campos *layer 3* (IP de origem e destino).



Agenda

1. Preparando o Ambiente
   1.1. Criando o Slice
   1.2. Criando Máquinas Virtuais
   1.3. Instalando o controlador kytos
   1.4. Definindo um *FlowSpace*

2. Executando o kytos
2.1. Primeiros passos
2.2. Instalando um NApp
2.3. Primeiro experimento com o Controlador Kytos
3. Experimento *Layer 3*
3.1. Criando um NApp
3.2. O NApp *Layer 3*
3.3. Testando o NApp *Layer 3*

Execução

1. Preparando o Ambiente

1.1. Criando o Slice

Visite o site https://ocf.fibre.org.br/ e realize login com seu usuário e senha.

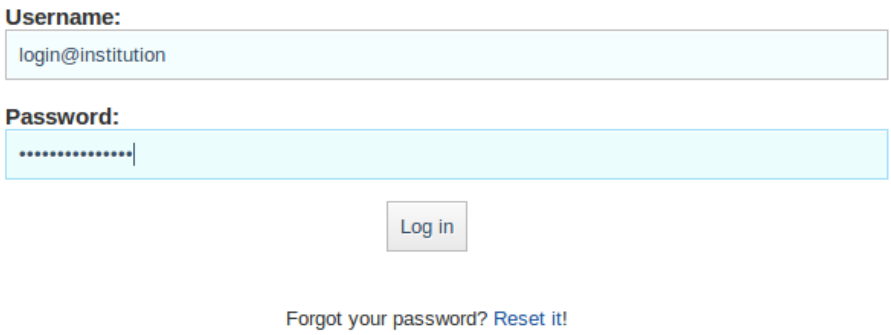


Figura 1

Na sua *dashboard*, selecione o projeto de interesse e crie um *slice* para realização do experimento. Se ainda não há um projeto criado, inicie a criação através do botão *"Create"* e preencha o formulário. No seu projeto e no seu *slice*, adicione os agregados de virtualização e *OpenFlow*, conforme a **figura 2**.

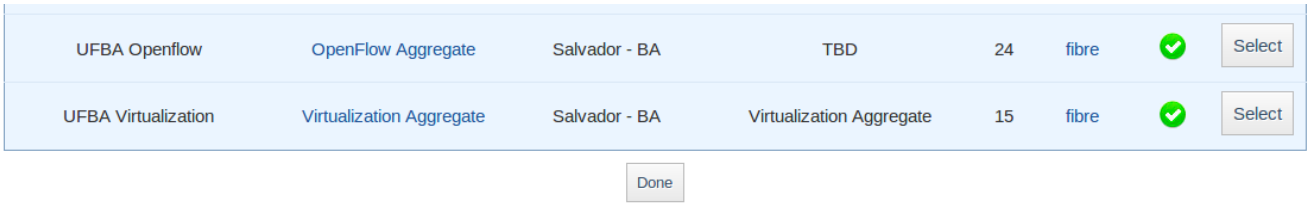


Figura 2

NOTA: É importante que nesse tutorial sejam utilizados Ilhas com *whitebox* e Debian 8 ou superior disponível, devido aos requisitos mínimos de funcionamento do controlador Kytos.

NOTA: Para a correta execução do experimento é necessário que hajam dois tipos de agregados e que pertençam à mesma ilha: *OpenFlow* e *Virtualization*

1.2. Criando Máquinas Virtuais

Seguindo a topologia proposta, iremos criar duas máquinas virtuais.

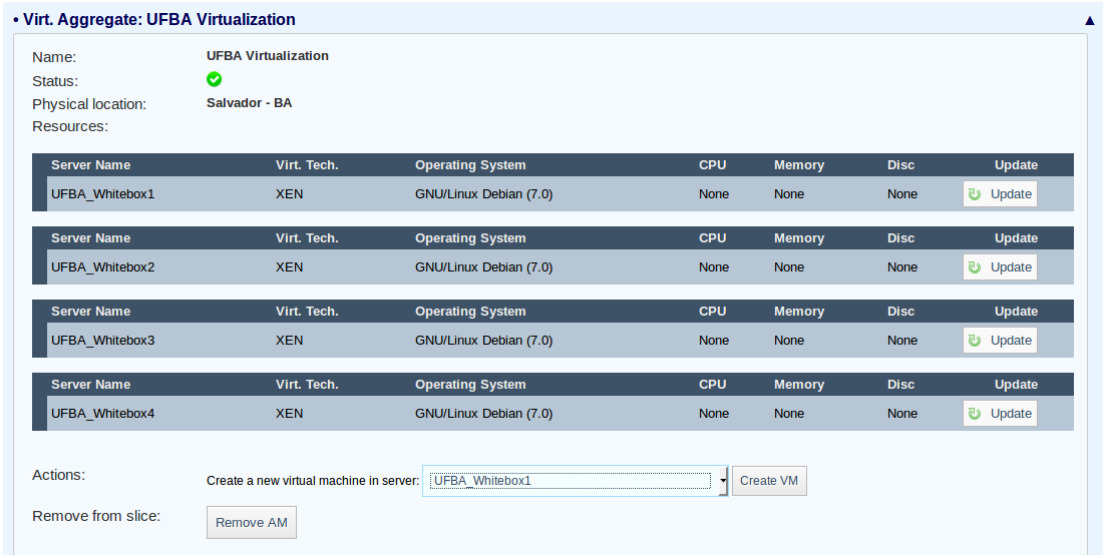


Figura 3

Navegue até a seção *Virt. Aggregate* (**Figura 3**). Note que na parte inferior há o campo *"Actions: Create a new virtual machine in server:*". Selecione um dos recursos disponíveis no menu e depois clique em "*Create VM*". Você será redirecionado para a seguinte página (**Figura 4**):

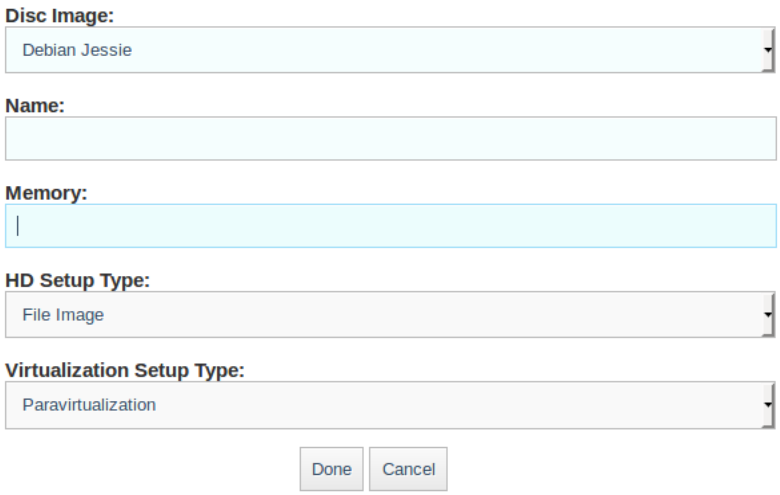


Figura 4

As máquinas necessárias para o projeto, deverão ser criadas com as seguintes especificações

Name	vm-controller	vm-host1
Disc Image	Debian Jessie	Debian Jessie
Memory	512	128
Hd Setup Type	File Image	File Image
Virtualization Setup Type	Paravirtualization	Paravirtualization

1.3. Instalando o Kytos

Kytos (https://kytos.io/) é um controlador SDN que opera através de aplicativos de rede (NApps, *Network Applications*), o que o faz funcionar de forma modular. Você utiliza e programa os NApps de acordo com seus interesses e necessidades da sua Rede.

O controlador Kytos ainda não é instalado por padrão nas máquinas virtuais do Fibre. Portanto, sua instalação faz-se necessária para o prosseguimento deste tutorial. Para instalar o controlador Kytos precisamos que a *vm-controller* tenha *Python 3.6* instalado nela. Nesta seção iremos instalar o python 3.6 e as dependências necessárias, antes de instalar o controlador kytos.

Voltando ao OCF, inicie a máquina virtual do controlador, apertando o botão *"Start"*:

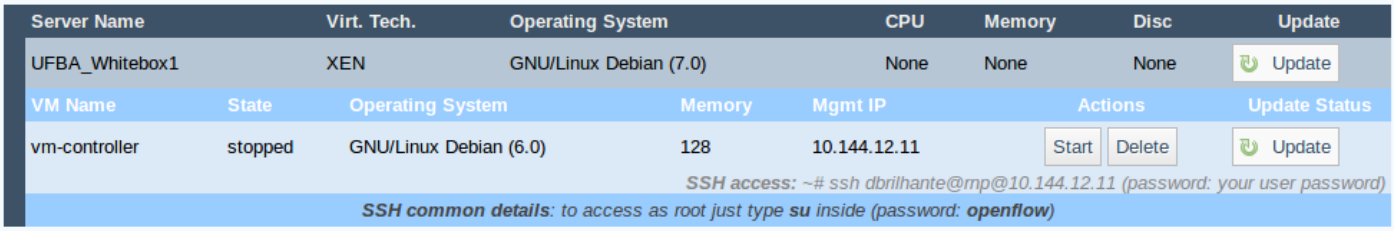


Figura 5

Quando a máquina terminar o processo de inicialização, realize SSH para ela com os seguintes dados: *seu_usuario@instituição@ip_vm-controller.* Quando realizar o login na máquina, obtenha permissões de superusuário executando o comando "*su*", com a senha **openflow**. Iniciaremos com a instalação das dependências:

```
# apt update
# apt upgrade
# apt install openssl ca-certificates python-openssl \
     pyca python3-openssl libssl-dev \
      build-essential checkinstall libreadline6-dev libncursesw5-dev \
      tk-dev libgdbm-dev libc6-dev libbz2-dev zlib1g-dev libsqlite3-dev \
      libncurses5-dev xz-utils tk-dev \
      libreadline-dev wget curl llvm git screen \
      liblzma-dev liblzma5 lzma lzma-dev python-lzma \
```

Se não houveram erros durante a instalação das dependências, passe para o próximo passo. Realize o download do Python 3.6:

```
#  wget https://www.python.org/ftp/python/3.6.1/Python-3.6.1.tgz
```

Vamos começar a instalar o Python 3.6

```
# tar xzf Python-3.6.1.tgz
# cd Python-3.6.1
```

```
# ./configure --enable-optimizations
# make -j8
```

Com o editor de texto de sua preferência (Nano, Vim, Emacs, etc), editaremos o arquivo Makefile, alterando a linha “CC= […]” para:

```
CC=         gcc -pthread -fprofile-arcs -ftest-coverage -lgcov --coverage
```

Execute o Makefile. Isso pode levar alguns minutos.

```
# make altinstall
```

Se a instalação foi bem sucedida, ao executar a linha seguinte você deverá acessar a CLI do Python 3.6

```
# python3.6
```

O Kytos será instalado em um ambiente virtual de desenvolvimento. Deste modo, a sua instalação estará segregada das bibliotecas Python padrão do seu sistema, facilitando os testes e a prototipação.

```
# python3.6 -m venv ~/test42
# source ~/test42/bin/activate
(teste42)# pip install --upgrade pip
```

Instale a última versão do Kytos disponível através do tutorial em https://tutorials.kytos.io/napps/development_environment_setup/#using-latest-kytos-from-github.

1.4. Definindo o FlowSpace

Dando continuidade ao nosso experimento, já temos as máquinas virtuais criadas, mas ainda não há conexão entre elas. Por isso, ainda não temos uma rede funcional, apenas máquinas virtuais isoladas. A comunicação entre essas duas máquinas depende da criação de uma topologia, neste caso, virtual.

A Rede do FIBRE é composta por duas redes distintas: a rede de controle e a rede de dados (ou de experimentação). As máquinas virtuais podem se comunicar através da rede de controle, mas para se comunicar através do plano de experimentação é necessário o um **controlador SDN**.

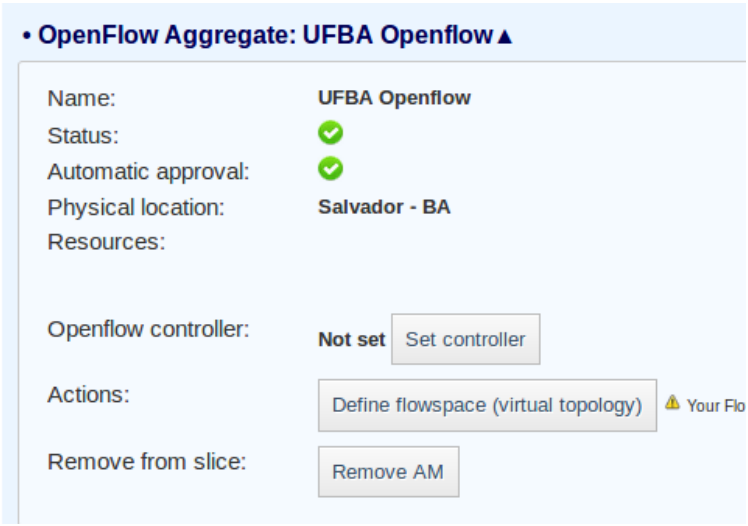


Figura 6

O controlador é o núcleo da rede SDN, ele define as regras que serão aplicadas aos *switches* SDN. Essas regras são estabelecidas através de uma aplicação que determina como a rede deverá se comportar.

Ao chegar um quadro a um switch, ele consulta o controlador para determinar que ação ele deverá tomar com aquele quadro. O controlador pode definir uma ação específica para quadros com as mesmas características.

O controlador já existe, na forma de uma máquina virtual, mas precisamos configurá-lo na rede.

Na seção "*OpenFlow Aggregate"* clique no botão "*Set controller".*

Você será redirecionado para uma página como a da **fig. 7**.

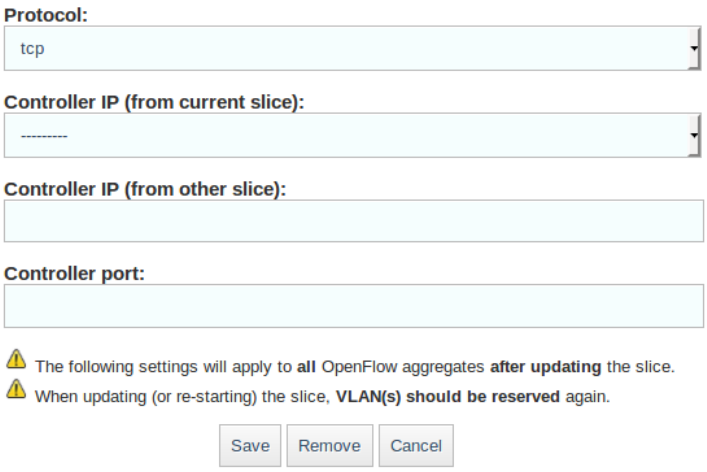


Figura 7

Preencha da seguinte forma e, ao terminar, clique no botão "*Save"*:

- *Protocol*: tcp
- *Controller IP (from current slice)*: vm-controller
- *Controller port*: 6633.

Definida a máquina virtual que irá hospedar o controlador para fazer a gestão do *slice*, podemos definir a topologia virtual. Como na **fig. 6**, clique em *"Define flowspace (virtual topology)"*. Você será redirecionado para a seguinte página:

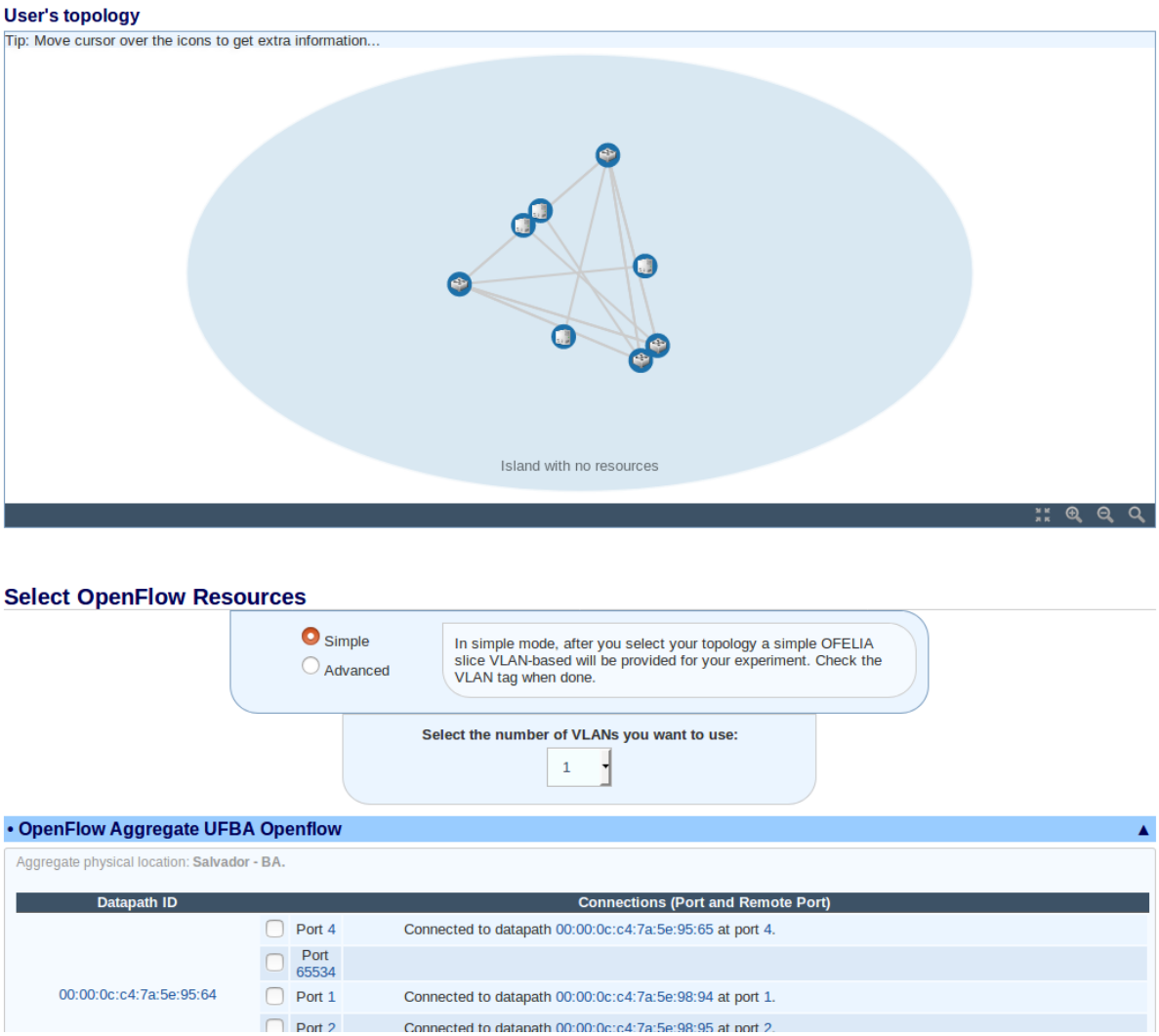


Figura 8

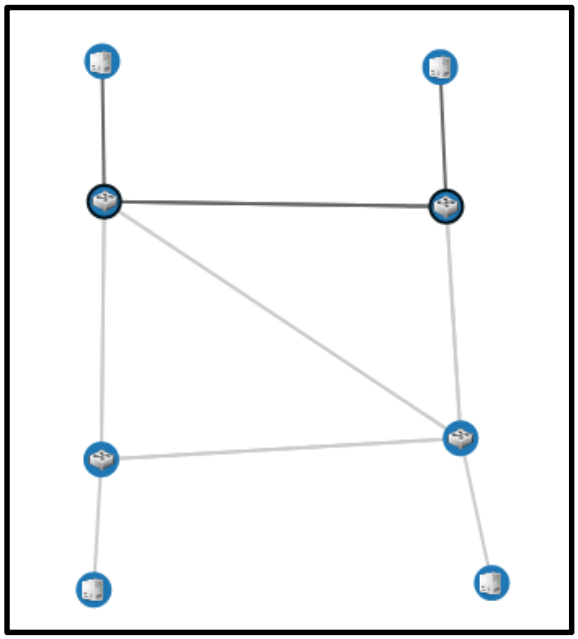


Figura 9

No quadro "*User's topology*", ao passar o mouse por cima dos elementos é possível ver informações sobre o seu slice. Procure os servidores que hospedam suas máquinas virtuais e selecione os enlace que os conectam aos *switches* e os enlaces que conectam os respectivos *switches* entre si. No fim sua topologia deverá se parecer com a da **fig. 8**.

Na parte *"Select OpenFlow Resources"* manteremos o número de VLANS necessários igual a 1. No caso de uma topologia mais complexa, poderíamos aumentar o número de VLANS.

Navegue até o fim da página e clique em *"Next"*. Você voltará à página inicial e então, pode acionar o botão "*Start slice"*. Não havendo erros, a primeira parte do tutorial foi concluída com sucesso.

2. Executando o Kytos

2.1. Primeiros Passos

Para utilizar o ambiente virtual instalado anteriormente faça:

```
# cd
# screen
# source ~/test42/bin/activate
```

No ambiente virtual, iremos ativar o controlador kytos:

```
# kytosd -f
```

Para abrir outra janela no *screen*, iremos utilizar 'ctrl+a c'. Na nova janela criada, iremos listar os NApps instalados.

```
# source ~/test42/bin/activate
# kytos napps list
```

2.2. Instalando um NApp

Antes de instalar o primeiro NApp, alguns pacotes são necessários:

```
# apt update
# apt install rrdtool librrd-dev
# pip install rrdtool
```

O kytos oferece um repositório com diversos NApps, que podem ser instalados para complementar seu controlador. Iremos buscar por *of_stats*

```
# kytos napps search of_stats
```

Se a coluna *status* mostra *[--]*, o NApp *of_stats* ainda não está instalado. Para instalar, basta executar o comando:

```
# kytos napps install kytos/of_stats
```

2.3. Primeiro experimento com o Controlador Kytos

A topologia de referência do nosso experimento compreende dois nós (vm-controller e vm-host1). Uma vez que a vm-controller já foi iniciada, repetiremos o mesmo processo para a vm-host1, assim como fizemos no passo 2.1 e realize o ssh para a vm-host1.

Para iniciar a topologia iremos configurar a VLAN em cada máquina virtual:

```
# su
# vconfig add eth1 <VLAN_ID>
# ifconfig eth1 up
```

NOTA: VLAN_ID foi definida junto com o flowspace e pode ser encontrado da seguinte forma:

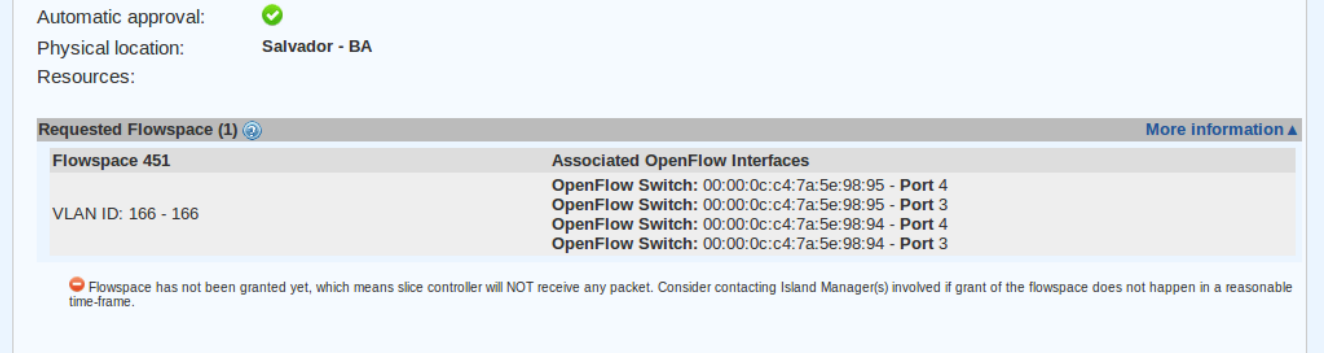


Figura 10

Na vm-controller, atribuiremos o seguinte IP:

```
# ifconfig eth1.<VLAN_ID> 192.168.10.10
```

E na vm-host1

```
# ifconfig eth1.<VLAN_ID> 192.168.10.20
```

Na vm-controller devemos ativar os NApps que irão fazer o controlador SDN assumir sua função e instalar os fluxos *OpenFlow* necessários.

```
# kytos napps enable kytos/of_core kytos/of_l2ls
```

Dispare um *ping* da vm-controller para a vm-host1. Se os pacotes forem transmitidos com sucesso, você concluiu mais uma etapa!

3. Experimento *Layer 3*

3.1. Criando um NApp

O Controlador Kytos permite a criação de módulos, os NApps, que adicionam funções customizadas ao controlador. Nesta parte do tutorial iremos aprender a criar um NApp.

Primeiro, iremos criar um repositório para os nossos NApps:

```
# mkdir ~/handson
# cd ~/handson
# kytos napps create
```

NOTA: Para executar a linha "`kytos napps create`" é preciso estar com o *virtual env* ativo.

Preencha o prompt da seguinte forma

```
Please, insert your NApps Server username: tutorial
Please, insert your NApp name: helloworld
Please, insert a brief description for your NApp [optional]: Hello, world!
```

O Kytos já criou para você um esboço do seu NApp, agora cabe a você transformá-lo em um NApp funcional!

```
# nano tutorial/helloworld/main.py
```

Seu NApp deve se parecer com:

```
from kytos.core import KytosNApp, log
from napps.tutorial.helloworld import settings


class Main(KytosNApp):

    def setup(self):
        pass

    def execute(self):
        pass

    def shutdown(self):
        pass
```

Acima das linhas "*pass*" adicione log.info("sua mensagem aqui!"). Seja criativo e tome cuidado com a identação. Salve o arquivo e saia. Uma sugestão de *Hello World* é:

```
from kytos.core import KytosNApp, log
from napps.tutorial.helloworld import settings


class Main(KytosNApp):

    def setup(self):
```

```
        log.info("Hello world! I am setting up")
        pass

    def execute(self):
        log.info("Hello world! I am executing up")
        pass

    def shutdown(self):
        log.info("Hello world! I am shutting down")
        pass
```

Edite também o arquivo "*kytos.json*", no mesmo diretório. Iremos alterar o campo version para "0.1.0"

```
{
  "author": "tutorial",
  "username": "tutorial",
  "name": "hellowolrd",
  "description": "Hello, World",
  "version": "0.1.0",
  "napp_dependencies": [],
  "license": "",
  "tags": [],
  "url": ""
}
```

Para instalar seu NApp, execute o comando a seguir

```
# kytos napps install tutorial/helloworld
```

Você deve estar no diretório *"handson"* para instalar seu NApp com sucesso. Agora, listando os NApps você deve visualizar seu NApp na lista de NApps instalados.

3.2. O NApp *Layer 3*

Repita os passos da seção 3.1 para a criação de um NApp e preencha o prompt da seguinte forma:

```
Please, insert your NApps Server username: tutorial
Please, insert your NApp name: of_l3ls
Please, insert a brief description for your NApp [optional]: This NApp does
packet switching using L3 information
```

Edite o arquivo *"main.py"* do NApp *"of_l3ls"* conforme descrito em https://tutorials.kytos.io/napps/switch_l3_part1/#final-main-py-file. Iremos adicionar regras openflow específicas para o funcionamento da NApp com o FIBRE, em **vermelho** no box abaixo. É necessário trocar **XXXX** pelo número de sua *vlan*, conforme definido na **seção 2.2**.

```
@listen_to('kytos/of_core.v0x01.messages.in.ofpt_packet_in')
    def handle_packet_in(self, event):
     #(…)

       if ethernet.ether_type.value == 0x800:
       #(...)
           if dest_port is not None:
                log.info('%s is at port %d.', ipv4.destination, dest_port)
                flow_mod = FlowMod()
                flow_mod.command = FlowModCommand.OFPFC_ADD
                flow_mod.match = Match()
                flow_mod.match.nw_src = ipv4.source
                flow_mod.match.nw_dst = ipv4.destination
                flow_mod.match.dl_type = ethernet.ether_type
                flow_mod.match.dl_vlan=XXXX
                flow_mod.idle_timeout = 10
                flow_mod.actions.append(ActionOutput(port=dest_port))
                event_out = KytosEvent(name=('tutorial/of_l3ls.messages.out.'
                                             'ofpt_flow_mod'),
                                       content={'destination': event.source,
                                                'message': flow_mod})
                self.controller.buffers.msg_out.put(event_out)
                log.info('Flow installed! Subsequent packets will be sent directly.')
                #(...)
```

Esse NApp atribui regras ao controlador que dão *match* pelos atributos: endereço de destino (*nw_dst*), endereço de origem (*nw_src*), tipo de pacote (*dl_type)* e tag de *vlan* (*dl_vlan*). A linha ***flow_mod.idle_timeout = 10*** irá configurar o seu controlador para que as regras instaladas permaneça por 10 segundos. Assim, a cada 10 segundos sem que hajam *matchs* nessa regra, os fluxos serão novamente instalados pelo controlador. Ao terminar, siga os passos da seção 3.1 para instalar seu NApp.

3.3. Testando o Napp *Layer 3*

Iremos testar a aplicação que acabamos de criar. Ao contrário do que se pode imaginar, um switch layer 3 não é capaz de realizar roteamento, mas sim de direcionar quadros para suas portas de acordo com o IP ao invés do MAC. Nossa aplicação é capaz de fazer isso, no entanto ela ainda não é capaz de preencher a tabela ARP de um switch.

Primeiro, desabilite o NApp *"of_l2ls"* e habilite o NApp *"of_l3ls"*

```
# kytos napps disable kytos/of_l2ls
# kytos napps enable tutorial/of_l3ls
```

Agora dispare um *ping* da vm-controller para vm-host1. Se as mensagens *echo-request* estiverem sendo respondidas com *echo-replies*, digite "*ctrl+a backspace"*, para visualizar o *prompt* do kytos. Você irá ver as mensagens de log definidas no NApp *"of_l3ls"*

```
2017-07-25 16:04:07,150 - INFO [tutorial/of_l3ls] (Thread-88) Packet received
from 10.0.0.1 to 10.0.0.2.
```

```
2017-07-25 16:04:07,165 - INFO [tutorial/of_l3ls] (Thread-90) Packet received
from 10.0.0.2 to 10.0.0.1.
2017-07-25 16:04:07,166 - INFO [tutorial/of_l3ls] (Thread-90) 10.0.0.1 is at
port 1.
2017-07-25 16:04:07,177 - INFO [tutorial/of_l3ls] (Thread-90) Flow installed!
Subsequent packets will be sent directly.
2017-07-25 16:04:08,148 - INFO [tutorial/of_l3ls] (Thread-94) Packet received
from 10.0.0.1 to 10.0.0.2.
2017-07-25 16:04:08,150 - INFO [tutorial/of_l3ls] (Thread-94) 10.0.0.2 is at
port 2.
2017-07-25 16:04:08,163 - INFO [tutorial/of_l3ls] (Thread-94) Flow installed!
Subsequent packets will be sent directly
```

Caso tenha ocorrido algum problema, confira o preenchimento da tabela ARP nas duas máquinas virtuais:

```
# arp
```

Ses o IPs associados às máquinas virtuais estiverem com estado "*incomplete*", você precisará preencher manualmente a tabela ARP, uma vez que este NApp ainda não suporta essa funcionalidade.

```
# arp -H ether -s <IP_remoteHost> <MAC_remoteHost> -i <iface_local>
```