

# Controle de congestionamento em TCP

Uma das funções principais do TCP é gerenciar o fluxo de mensagens entre origem e destino, adaptando a taxa de transmissão da origem à taxa de recepção no destino de maneira a procurar manter a melhor taxa de transmissão possível e não sobrecarregar a rede ou o destinatário. O objetivo do experimento descrito aqui é que o aluno entenda a necessidade de controle de congestionamento, assim como o funcionamento básico dos algoritmos usados pelo TCP.

O estudante deve poder controlar as condições da rede durante a execução do experimento para observar como cada algoritmo se comporta. Para exercer esse controle, o aluno pode usar uma máquina intermediária que reencaminha mensagens de uma máquina origem para uma máquina destino introduzindo atrasos, gargalos, ou a perda de pacotes aleatórios. A Figura 1 mostra a topologia dessa rede virtual.

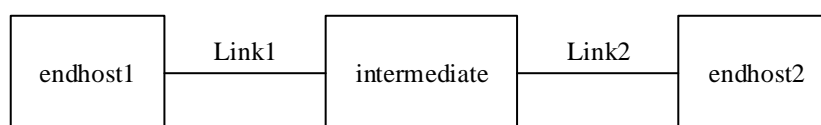


Figura 1. Topologia usada no experimento

Após alocar os recursos necessários, o estudante deve então: (1) configurar a máquina intermediária para reencaminhar pacotes introduzindo atrasos ou simulando uma determinada banda; (2) prepara o ambiente para captura de variáveis internas do TCP na máquina origem e destino; (3) gerar o tráfego na máquina origem e (4) analisar os resultados capturados. O aluno deverá repetir o experimento variando os atrasos e a banda simulada.

Na seção seguinte, explicamos como preparar o ambiente da Figura 1 na infraestrutura do testbed FIBRE.

## Configuração do ambiente

Essa seção descreve como configurar o ambiente para o experimento construindo uma rede virtual dedicada. Utilizaremos o framework de controle OCF, um dos frameworks disponíveis no portal FIBRE.

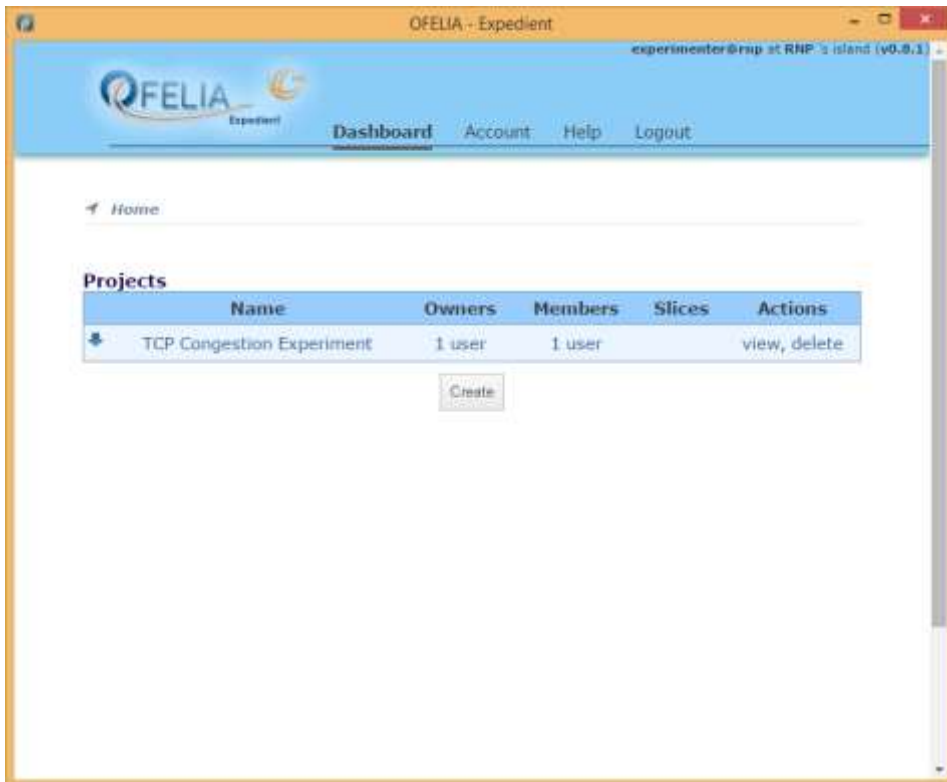
O controle de acesso do OCF é associado ao conceito de projeto. Cada projeto tem a seu dispor um determinado conjunto de recursos e fica sob a coordenação de um pesquisador principal, que pode alocar permissões a outros usuários.

Uma vez obtidas as permissões em um projeto, o usuário pode começar a preparar o experimento. O OCF utiliza o conceito de slice para descrever um experimento. Um slice descreve um conjunto de recursos e as configurações associadas a eles, assim como o estado do experimento. Ao criar um novo slice, um usuário do FIBRE deve fornecer informações como nome, descrição e data de expiração (tempo de vida do slice). Depois da expiração do slice, o administrador pode liberar os recursos associados a ele.

Após, o usuário deve adicionar recursos ao slice. Os recursos são representados por agregados (um agregado pode ser um rack com várias máquinas, por exemplo). O usuário escolhe os agregados necessários nas ilhas desejadas.

**Passo a passo:**

1 – Escolha o projeto de seu experimento, ou solicite a criação de um novo projeto clicando no botão Create da interface de projetos do OCF.



2 – Crie um novo slice dentro do projeto de seu experimento através do botão *Create slice*:



3 – Dê um nome e uma descrição ao seu *Slice* e clique em *Save*:

OFELIA - Create slice

experimenter@rnp at RNP's island (v0.8.1)

Dashboard Account Help Logout

Home > Project TCP Congestion Experiment

### Create Slice in Project TCP Congestion Experiment

Name:  
Experiment 1

Description:  
My first experiment

Save Cancel

4 – Adicione os agregados OpenFlow e virtualização ao seu projeto (botão *Add Aggregates* na interface do projeto):

OFELIA - Aggregates for Project TCP Congestion Experiment

experimenter@rnp at RNP's island (v0.8.1)

Dashboard Notifications Account Help Logout

Home > Project TCP Congestion Experiment > Add Project Aggregates

Successfully created slice Experiment 1.

### Add an Aggregate for Project TCP Congestion Experiment

Name	Type	Location	Description	Size	Managers	Status	Actions
RNP Openflow	OpenFlow Aggregate	Brasilia	Openflow agregate of RNP island	27	root	✓	Select
RNP Virtualization	Virtualization Aggregate	Brasilia	Virtualization agregate of RNP island	20	root	✓	Select

Done

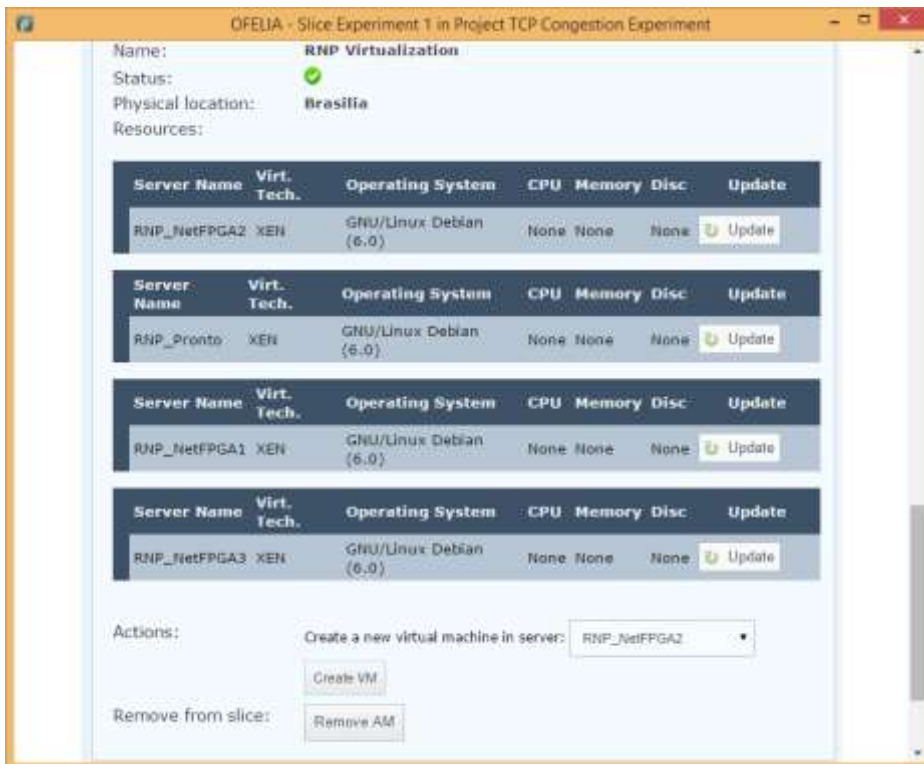
5 – Entre na interface de seu *slice* e adicione os agregados do projeto (botão *Add an Aggregate Manager to the current slice*):



6 – Verifique a topologia do seu *slice*, certificando-se que todos os componentes estejam interconectados:



7 – Crie máquinas virtuais para seu experimento em servidores de virtualização diferentes. Selecione o servidor escolhido e clique no botão *Create VM*:



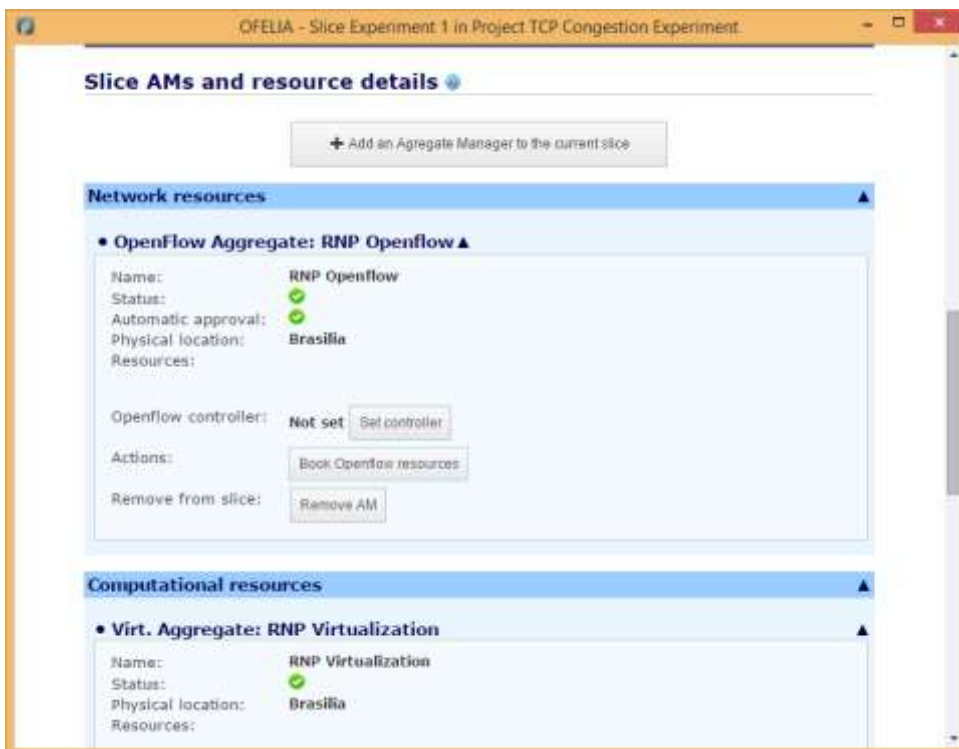
8 – Dê um nome a sua máquina virtual. Nesse exemplo criaremos três máquinas virtuais: *endhost1*, *endhost2* e *intermediate*. A opção *Disk Image* deve estar marcada como *Default* com 128MB de RAM para cada uma.



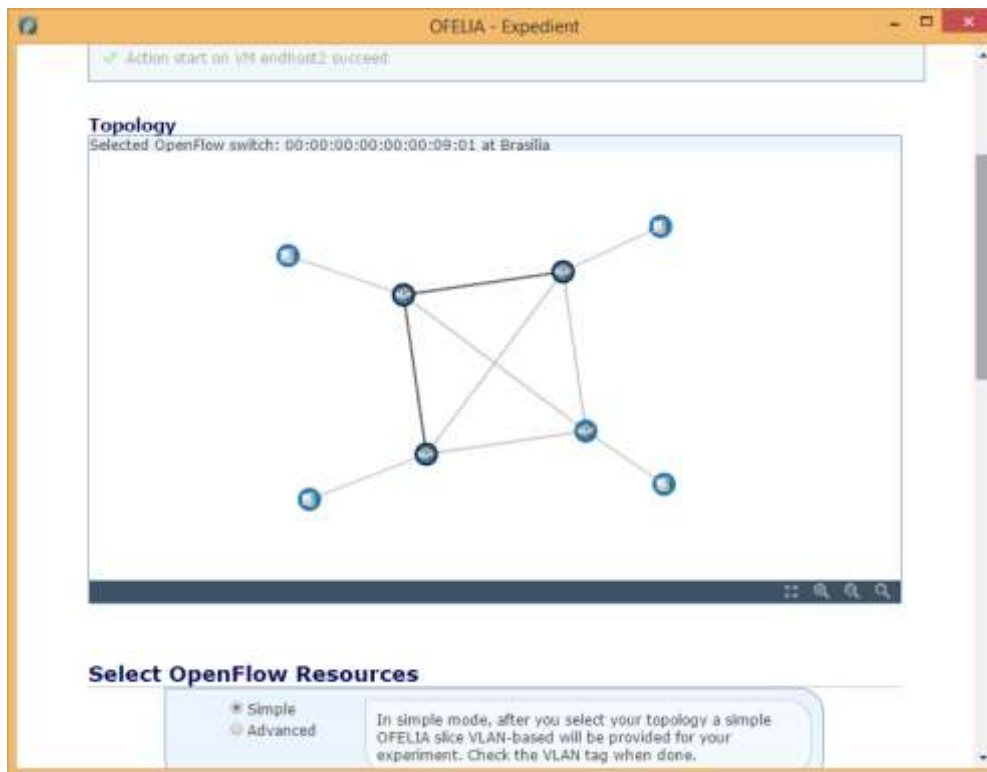
9 – Após a criação das máquinas virtuais, anote os IPs alocados para cada uma e as inicie através dos botões *Start*:



10 – No próximo passo, o usuário deve alocar os recursos necessários a seu experimento. As ações “Set Controller” e “Book Openflow Resources” na área de gerenciamento de recursos de redes permitem que o usuário especifique o endereço do controlador *Openflow* do experimento e o(s) *flowspace(s)* a serem requisitados. Aloque os recursos de rede para seu experimento através do botão *Book Openflow resources* na área *Network resources*:



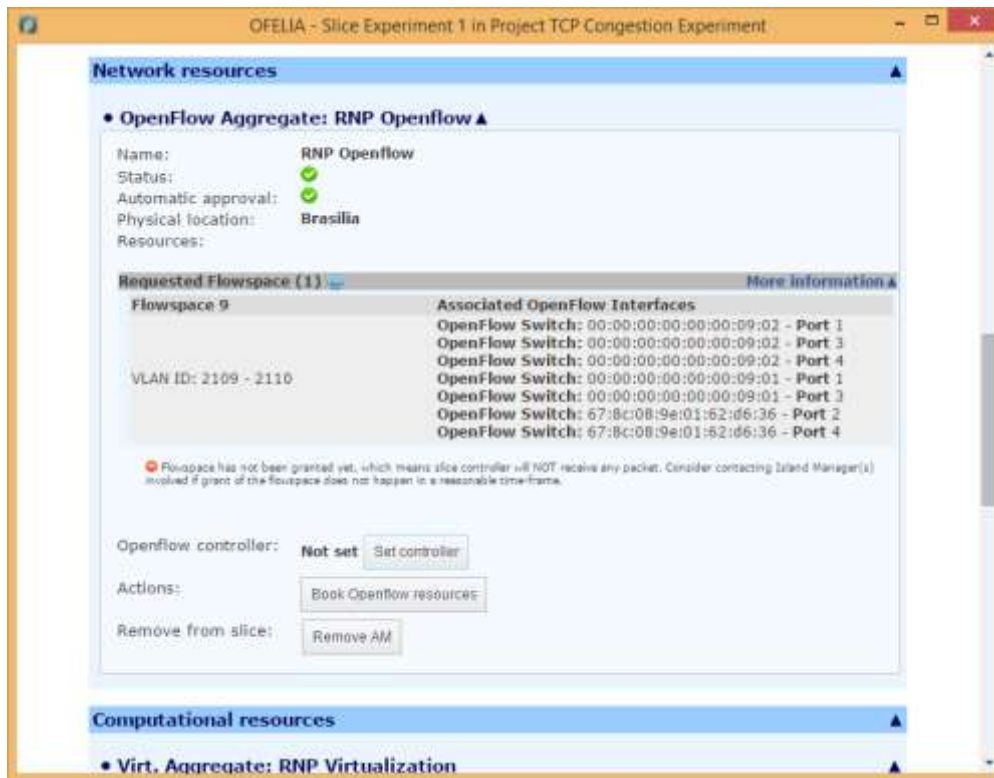
11 – Selecione o caminho de rede desejado que liga todas as VMs na área Topology:



12 – Selecione a opção do modo de alocação simples (*Simple*) e solicite a alocação de duas VLANs para seu experimento:

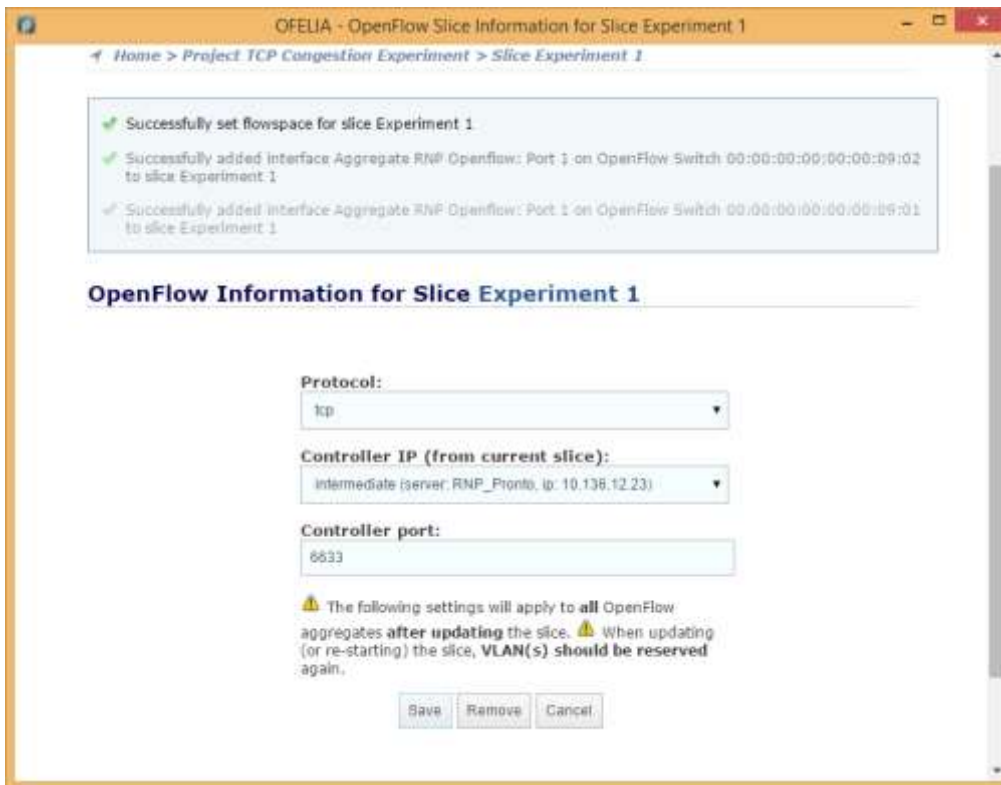


13 – Clique na opção *More Information* na área Network resources e anote o número das VLANs alocadas para seu experimento:

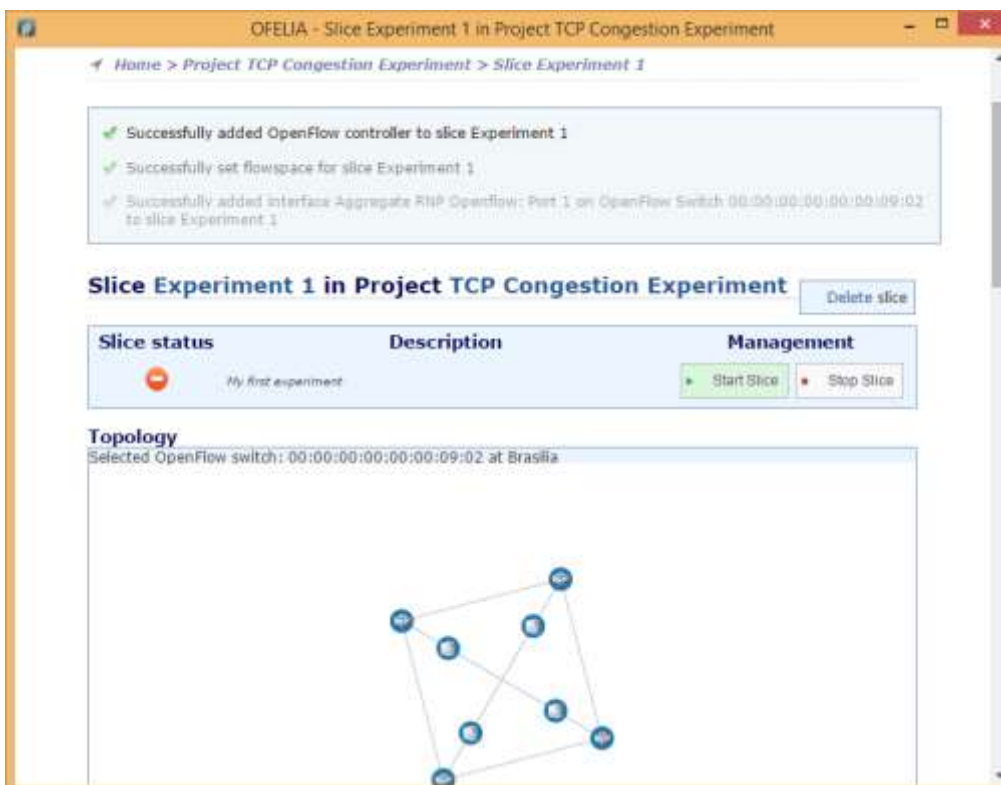


14 – Uma vez especificados os *flowspace*s e criadas as VMs, o usuário deve definir o endereço IP do controlador para ser o mesmo da VM que hospeda o controlador OpenFlow. Clique no botão *Set controller* e escolha uma máquina virtual do *slice* para hospedar o controlador *Openflow* do experimento, nesse exemplo, utilizamos a máquina virtual *intermediate* e a porta 6633:





15 – Para executar o experimento, o usuário inicia o *slice* na página de gerenciamento de *slices* através do botão *Start slice*:

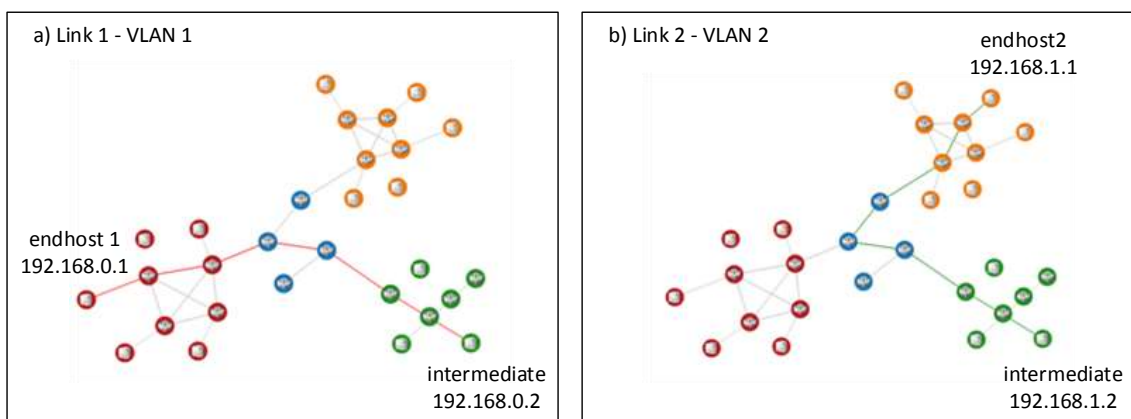


16 – Verifique se o *slice* foi iniciado com sucesso:



### Execução do experimento

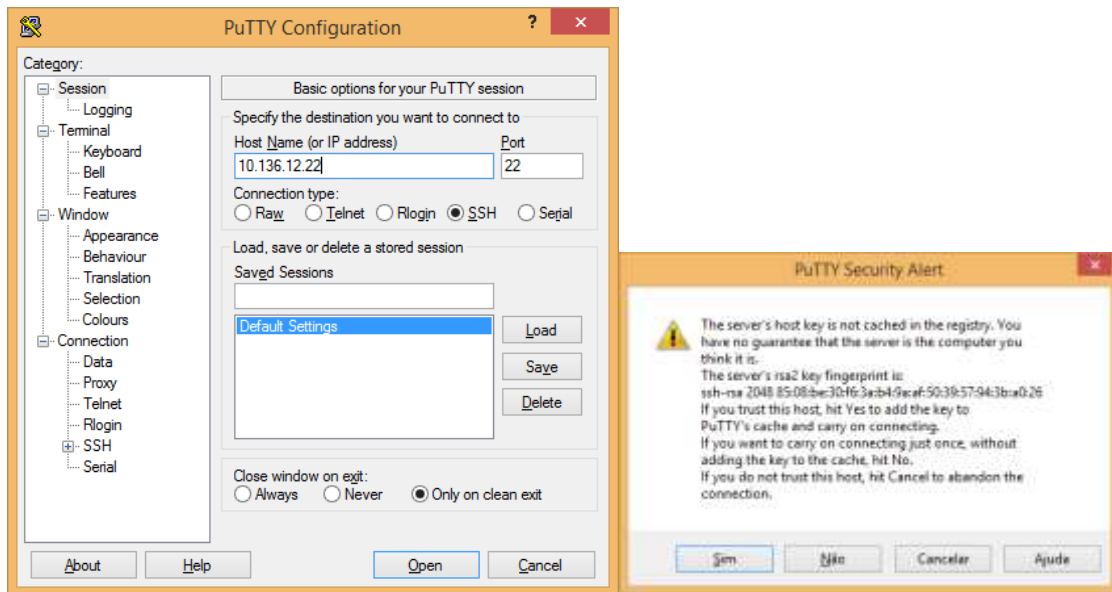
Depois de configurado o ambiente, o usuário deve acessar cada VM para configurar a rede virtual alocada ao experimento. Neste caso, alocamos três máquinas virtuais (*endhost1*, *endhost2*, e *intermediate*) e dois *flowspaces*, um entre *endhost1* e *intermediate* e outro entre *intermediate* e *endhost2*, mostrados na figura abaixo.



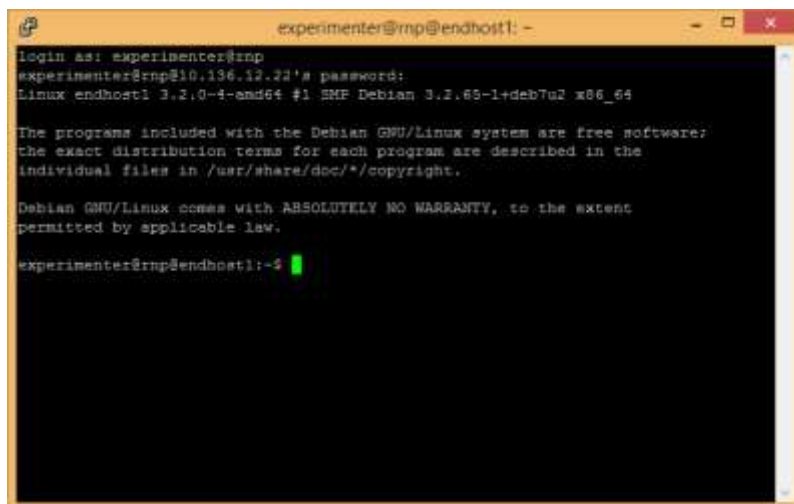
1.

A topologia da rede virtual do experimento.

Uma vez que o slice esteja ativo, o experimentador pode usar as VMs como máquinas remotas, por ssh. Usando ssh e scp, o usuário pode instalar qualquer software necessário a seu experimento. Inicie seu *slice*.



Para acessar as VMs é necessário estar conectado à VPN de experimento. A autenticação na VM é realizada usando o mesmo usuário e senha utilizado para logar-se ao OCF.



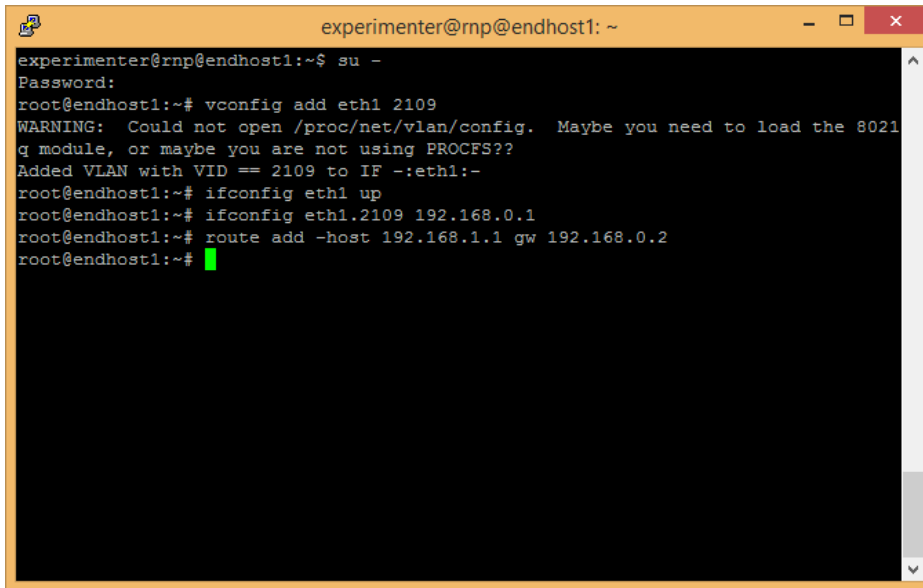
No próximo passo, o usuário deve acessar cada VM, via ssh para o IP fornecido, e configurar as redes virtuais (VLANs) e interfaces de rede para o experimento. As figuras abaixo mostram a configuração das VLANs e interfaces de rede das VMs *endhost1* e *endhost2*. Em cada máquina, o usuário usa o comando `vconfig` para criar uma nova rede virtual (com id 2109 e 2110 respectivamente) associada à interface de rede `eth1`, e a seguir usa o comando `ifconfig` para habilitar essa interface. Finalmente, usa o comando `ifconfig` mais uma vez para configurar o endereço IP associado a essa interface e o comando `route` para adicionar a rota para o outro *endhost* via a máquina intermediária. Os números de VLAN deverão ser substituídos pelas VLANs fornecidos para seu experimento.

```

root@endhost1:~# vconfig add eth1 2109
Added VLAN with VID == 2109 to IF -:eth1:-
root@endhost1:~# ifconfig eth1 up
root@endhost1:~# ifconfig eth1.2109 192.168.0.1
root@endhost1:~# route add -host 192.168.1.1 gw 192.168.0.2

```

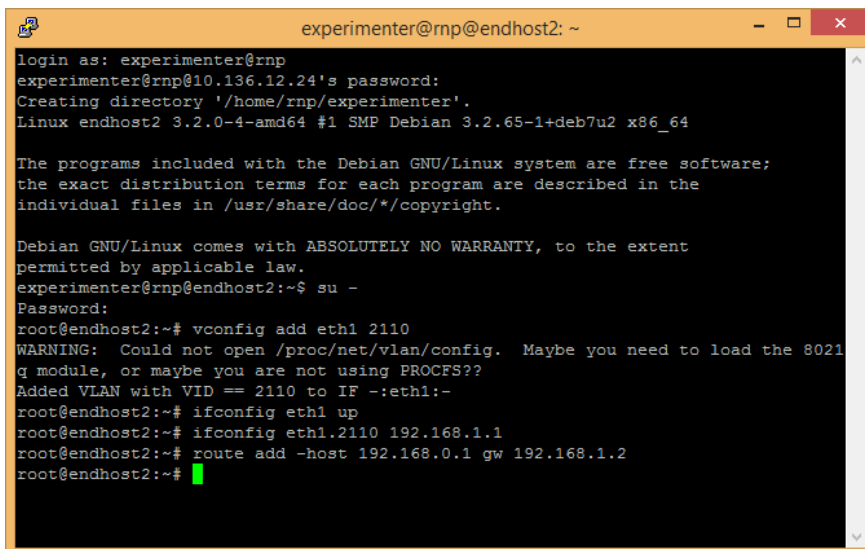
### Configuração de *endhost1*.



```
experimenter@rnp@endhost1: ~  
experimenter@rnp@endhost1:~$ su -  
Password:  
root@endhost1:~# vconfig add eth1 2109  
WARNING: Could not open /proc/net/vlan/config. Maybe you need to load the 8021  
q module, or maybe you are not using PROCFS??  
Added VLAN with VID == 2109 to IF -:eth1:-  
root@endhost1:~# ifconfig eth1 up  
root@endhost1:~# ifconfig eth1.2109 192.168.0.1  
root@endhost1:~# route add -host 192.168.1.1 gw 192.168.0.2  
root@endhost1:~# █
```

```
root@endhost2:~# vconfig add eth1 2110  
Added VLAN with VID == 2110 to IF -:eth1:-  
root@endhost2:~# ifconfig eth1 up  
root@endhost2:~# ifconfig eth1.2110 192.168.1.1  
root@endhost2:~# route add -host 192.168.0.1 gw 192.168.1.2
```

### Configuração de *endhost2*.



```
experimenter@rnp@endhost2: ~  
login as: experimenter@rnp  
experimenter@rnp@10.136.12.24's password:  
Creating directory '/home/rnp/experimenter'.  
Linux endhost2 3.2.0-4-amd64 #1 SMP Debian 3.2.65-1+deb7u2 x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
experimenter@rnp@endhost2:~$ su -  
Password:  
root@endhost2:~# vconfig add eth1 2110  
WARNING: Could not open /proc/net/vlan/config. Maybe you need to load the 8021  
q module, or maybe you are not using PROCFS??  
Added VLAN with VID == 2110 to IF -:eth1:-  
root@endhost2:~# ifconfig eth1 up  
root@endhost2:~# ifconfig eth1.2110 192.168.1.1  
root@endhost2:~# route add -host 192.168.0.1 gw 192.168.1.2  
root@endhost2:~# █
```

A figura abaixo mostra a configuração da máquina intermediária. Inicialmente, o usuário usa o comando `vconfig` para configurar as duas redes virtuais ligadas a ambos *endhosts* e configura os endereços IP de cada rede através do comando `ifconfig`. Em seguida, o usuário deve habilitar o encaminhamento de pacotes entre as duas redes através dos comandos `sysctl` e `iptables`. Finalmente, atrasos ou gargalos artificiais podem ser introduzidos usando a ferramenta de controle de tráfego `tc`. Nesse exemplo, configuramos um atraso artificial de 150ms para cada pacote repassado da máquina *endhost1* para o *endhost2*.

```
root@intermediate:~# vconfig add eth1 2109  
Added VLAN with VID == 2109 to IF -:eth1:-
```

```

root@intermediate:~# ifconfig eth1 up
root@intermediate:~# ifconfig eth1.2109 192.168.0.2
root@intermediate:~# vconfig add eth1 2110
Added VLAN with VID == 2110 to IF -:eth1:-
root@intermediate:~# ifconfig eth1.2110 192.168.1.2
root@intermediate:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@intermediate:~# tc qdisc add dev eth1.2110 root netem delay 150ms

```

### Configuração da máquina intermediária.

```

experimenter@rnp@intermediate: ~
experimenter@rnp@intermediate:~$ su -
Password:
root@intermediate:~# vconfig add eth1 2109
Added VLAN with VID == 2109 to IF -:eth1:-
root@intermediate:~# ifconfig eth1 up
root@intermediate:~# ifconfig eth1.2109 192.168.0.2
root@intermediate:~# vconfig add eth1 2110
Added VLAN with VID == 2110 to IF -:eth1:-
root@intermediate:~# ifconfig eth1.2110 192.168.1.2
root@intermediate:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@intermediate:~# tc qdisc add dev eth1.2110 root netem delay 150ms
root@intermediate:~# █

```

Finalmente, inicie o controlador OpenFlow na máquina intermediária. A imagem padrão fornece o controlador POX por padrão. Para iniciar digite *pox* na linha de comando:

```

experimenter@rnp@intermediate: ~
experimenter@rnp@intermediate:~$ su -
Password:
root@intermediate:~# vconfig add eth1 2109
Added VLAN with VID == 2109 to IF -:eth1:-
root@intermediate:~# ifconfig eth1 up
root@intermediate:~# ifconfig eth1.2109 192.168.0.2
root@intermediate:~# vconfig add eth1 2110
Added VLAN with VID == 2110 to IF -:eth1:-
root@intermediate:~# ifconfig eth1.2110 192.168.1.2
root@intermediate:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@intermediate:~# tc qdisc add dev eth1.2110 root netem delay 150ms
root@intermediate:~# pox
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
[core ] POX 0.3.0 (dart) is up.
[openflow.of_01 ] [08-9e-01-62-d6-36|26508 3] connected
[openflow.of_01 ] [00-00-00-00-09-01 2] connected
[openflow.of_01 ] [00-00-00-00-09-02 1] connected
█

```

Uma vez que o ambiente tenha sido configurado, a configuração do experimento pode ser verificada através da medição do atraso entre as máquinas virtuais com o comando *ping* a partir da máquina virtual *endhost1* para a máquina *endhost2* utilizando a máquina *intermediate* para o roteamento:

```
experimenter@mp@endhost1: ~  
root@endhost1:~# ping 192.168.0.2  
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.  
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=68.3 ms  
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.196 ms  
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.213 ms  
^C  
--- 192.168.0.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.196/22.913/68.331/32.115 ms  
root@endhost1:~# ping 192.168.1.1  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_req=1 ttl=63 time=1468 ms  
64 bytes from 192.168.1.1: icmp_req=2 ttl=63 time=470 ms  
64 bytes from 192.168.1.1: icmp_req=3 ttl=63 time=150 ms  
64 bytes from 192.168.1.1: icmp_req=4 ttl=63 time=150 ms  
64 bytes from 192.168.1.1: icmp_req=5 ttl=63 time=150 ms  
^C  
--- 192.168.1.1 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4009ms  
rtt min/avg/max/mdev = 150.439/478.169/1468.625/510.543 ms, pipe 2  
root@endhost1:~# █
```

Ao terminar o experimento, o usuário deve liberar os recursos utilizados. Para isso, basta selecionar o botão "Stop Slice" área de gerenciamento de slices (Figura 2). Essa ação libera todos os recursos alocados ao experimento.