Mininet- Criando Topologias

OIRC - Interconexão de redes de computadores
Prof. Dr. Ricardo José Pfitscher
ricardo.pfitscher@gmail.com

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Objetivos de aprendizagem

- Entender como criar uma topologia específica no Mininet

Cronograma

- Walkthrough mininet - fechamento
- Criando uma topologia simples
- Geração de carga de trabalho controlada

Walkthrough

- O objetivo do walkthrough era se familiarizar com a ferramenta
- Considerações:
  - Cada componente executa em um *container* diferente
  - É possível executar comandos pré-definidos ou comandos quaisquer do linux dentro de cada container
    - Ex.: h1 ping h2, pingall, h1 top
  - Na parte de “Custom Topologies” o tutorial ensina a utilizar uma topologia custom
  - Também é possível utilizar MACs simplistas
    - Ex.: HWaddr 00:00:00:00:00:01
    - Isso pode facilitar a vida quando formos pensar na lógica do controlador

Criando topologias

- O comando abaixo permite apontar para um script em python que cria uma topologia
  - `sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo --test pingall`

Se você tirar o --test ele abre o cli

- Você pode criar uma cópia desse arquivo através de:
  - `cd ~/mininet/custom/`
  - `cp topo-2sw-2host.py minha-topo.py`
- Você pode editar essa topologia com:
  - `vim minha-topo.py`
- Ou codificar fora da VM e copiar para a VM com pscp.exe
  - `pscp.exe minha-topo.py mininet@IP_DA_VM:/home/mininet/custom`

Comandos úteis:
i → inicia modo edição
esc → volta para comandos
:q → sai sem salvar
:wq → salva e sai
:w → salva

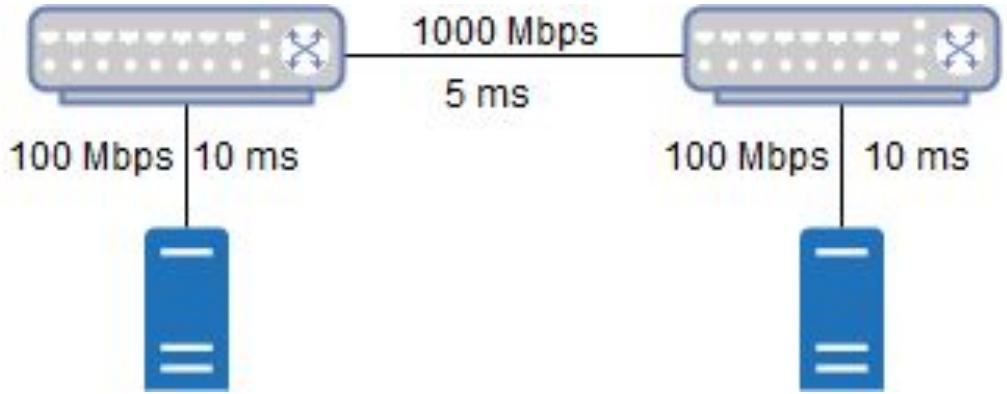
```
from mininet.topo import Topo
class MyTopo( Topo ):
    def __init__( self ):
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )
        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Topologias avançadas

- É possível configurar atraso e largura de banda entre os links
  - `from mininet.link import TCLink`
  - `…`
  - `self.addLink(leftSwitch, rightSwitch , cls=TCLink, bw=1000, delay='10ms')`
- É possível limitar a cpu e definir o IP dos hosts
  - `from mininet.node import CPULimitedHost`
  - `leftHost = self.addHost( 'h1',cpu=.5, ip='10.0.0.1')`

Topologias avançadas

- Crie a seguinte topologia e verifique se o delay configurado e a largura de banda são alcançados
  - Utilize ping e iperf para medir, respectivamente delay e throughput

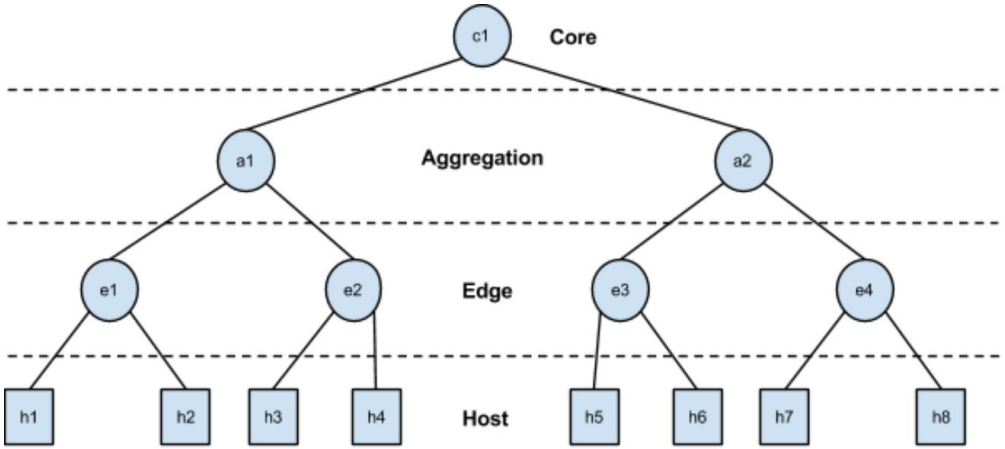


Experimentos automatizados

- É possível executar as topologias de forma automatizada
  - **Vejamos o exemplo no Moodle**
- Desenhe a topologia resultante
- Faça os testes de largura de banda e atraso entre todos os hosts

Exercícios

- Construa a seguinte topologia no mininet e faça os testes de conectividade e largura de banda
  - digite o comando *net* no mininet para ver a rede construída



Experimentos automatizados [2]

- Não é obrigatório utilizar o comando mn para iniciar o mininet
- Você pode chamar as funções do mininet diretamente de um programa em python
  - Vejamos o exemplo 2 no **Moodle**
- Para executar esse exemplo é necessário um **controlador externo**
  - `c0 = RemoteController('c0', ip='127.0.0.1')`
- O controlador pode estar na mesma VM (127.0.0.1) ou em outra (**ideal**)
- Nessa aula vamos utilizar o POX
  - A VM do mininet já vem com esse controlador instalado!
  - Muitas infos aqui: https://openflow.stanford.edu/display/ONL/POX+Wiki

Experimentos automatizados [2]

- Utilize duas conexões SSH (putty.exe) até a VM do mininet
  - Uma para o controlador
  - Outra para o ambiente de rede
- Na conexão para o controller, digite:

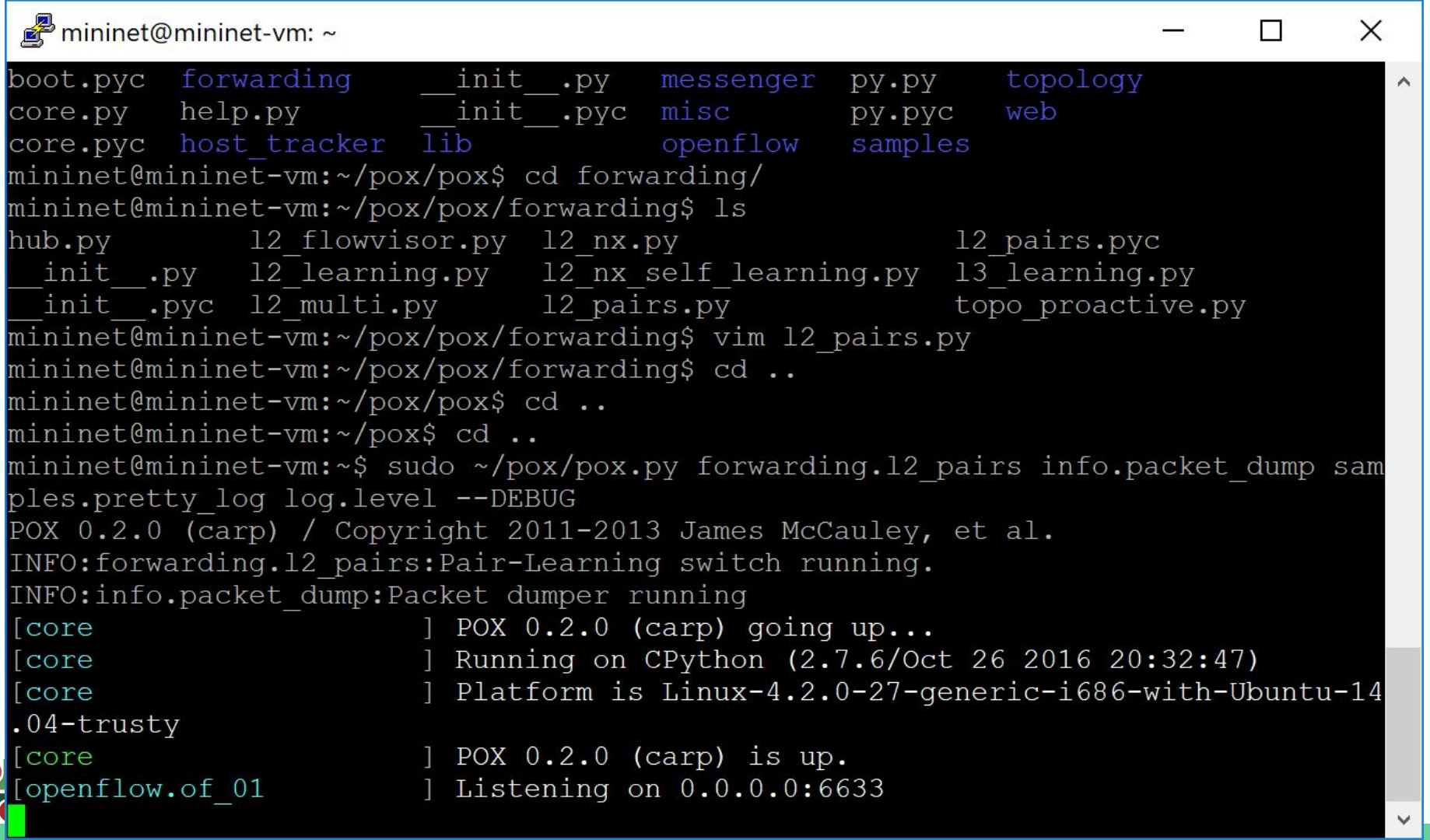
```
sudo ~/pox/pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log
log.level --DEBUG
```

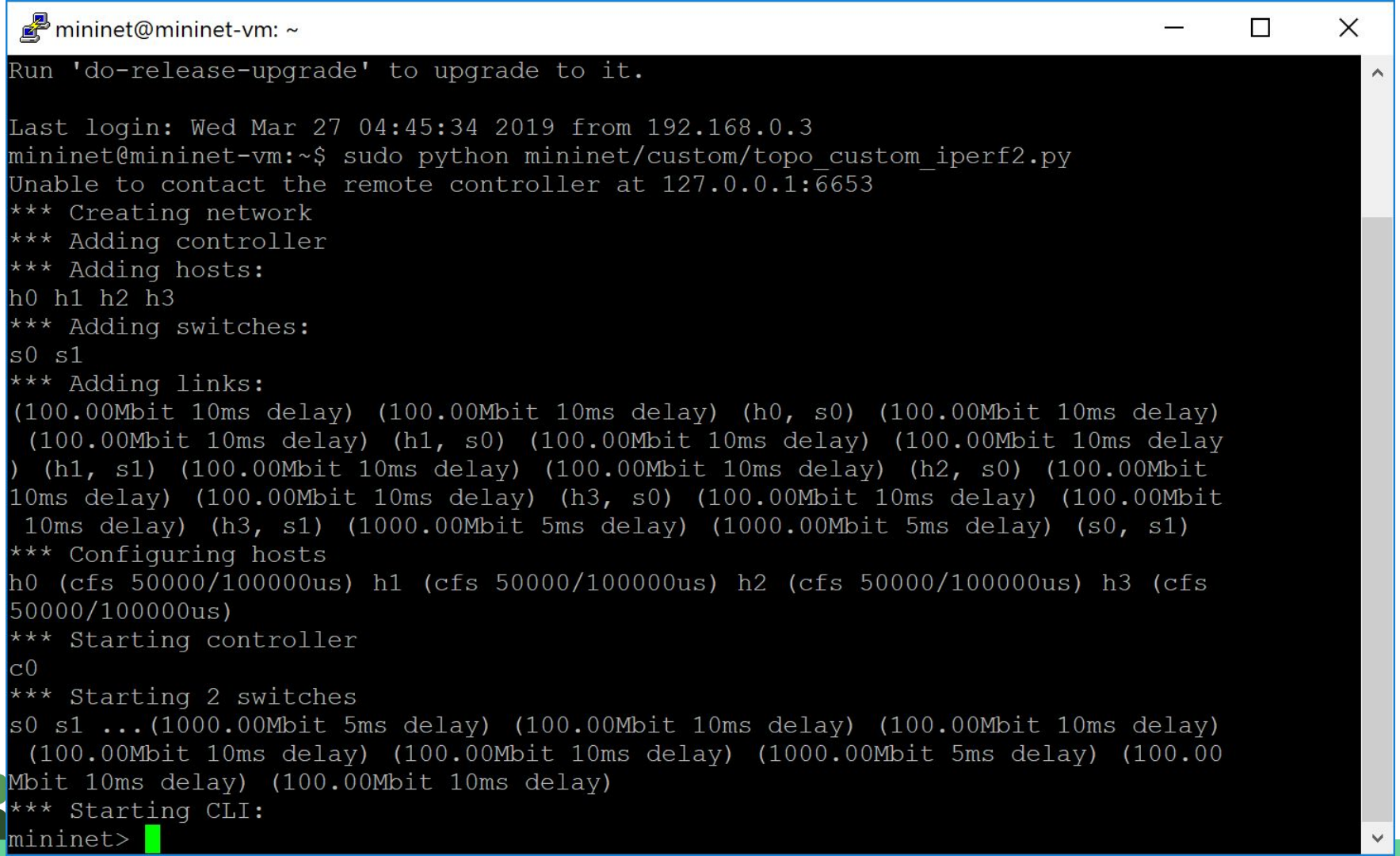
- Copie o código do exemplo2 para o ambiente do mininet
- No cmd do windows digite (pscp deve estar no diretório):

```
pscp topo_custom_iperf2.py
mininet@192.168.0.11:/home/mininet/mininet/custom/
```

- Na conexão para o ambiente de rede, digite:

```
sudo python mininet/custom/topo_custom_iperf2.py
```


mininet@mininet-vm: ~
boot.pyc forwarding __init__.py messenger py.py topology
core.py help.py __init__.pyc misc py.pyc web
core.pyc host_tracker lib openflow samples
mininet@mininet-vm:~/pox/pox$ cd forwarding/
mininet@mininet-vm:~/pox/pox/forwarding$ ls
hub.py l2_flowvisor.py l2_nx.py l2_pairs.pyc
__init__.py l2_learning.py l2_nx_self_learning.py l3_learning.py
__init__.pyc l2_multi.py l2_pairs.py topo_proactive.py
mininet@mininet-vm:~/pox/pox/forwarding$ vim l2_pairs.py
mininet@mininet-vm:~/pox/pox/forwarding$ cd ..
mininet@mininet-vm:~/pox/pox$ cd ..
mininet@mininet-vm:~/pox$ cd ..
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_pairs info.packet_dump sam
ples.pretty_log log.level --DEBUG
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
INFO:info.packet_dump:Packet dumper running
[core] POX 0.2.0 (carp) going up...
[core] Running on CPython (2.7.6/Oct 26 2016 20:32:47)
[core] Platform is Linux-4.2.0-27-generic-i686-with-Ubuntu-14
.04-trusty
[core] POX 0.2.0 (carp) is up.
[openflow.of_01] Listening on 0.0.0.0:6633


mininet@mininet-vm: ~
Run 'do-release-upgrade' to upgrade to it.
Last login: Wed Mar 27 04:45:34 2019 from 192.168.0.3
mininet@mininet-vm:~$ sudo python mininet/custom/topo_custom_iperf2.py
Unable to contact the remote controller at 127.0.0.1:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h0 h1 h2 h3
*** Adding switches:
s0 s1
*** Adding links:
(100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h0, s0) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h1, s0) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h1, s1) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h2, s0) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h3, s0) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (h3, s1) (1000.00Mbit 5ms delay) (1000.00Mbit 5ms delay) (s0, s1)
*** Configuring hosts
h0 (cfs 50000/100000us) h1 (cfs 50000/100000us) h2 (cfs 50000/100000us) h3 (cfs 50000/100000us)
*** Starting controller
c0
*** Starting 2 switches
s0 s1 ...(1000.00Mbit 5ms delay) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay) (1000.00Mbit 5ms delay) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay)
*** Starting CLI:
mininet>

Experimentos automatizados [2]

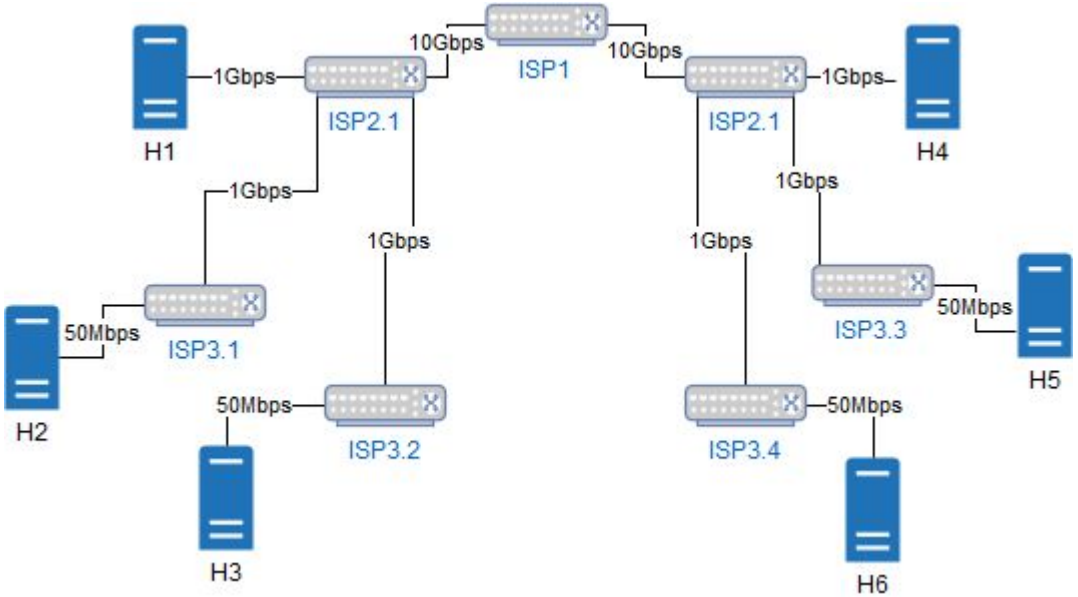
- Faça os testes de conectividade e largura de banda
  - Observe as ações do controlador
- Verifique o que acontece se você executar a topologia sem o controlador
  - Faça um pingall

Gerando a carga de trabalho automática

- Quando executamos o programa em python que gera a topologia, é possível criar funções para executar comandos nos hosts
- Vejamos o exemplo 3 no **Moodle**

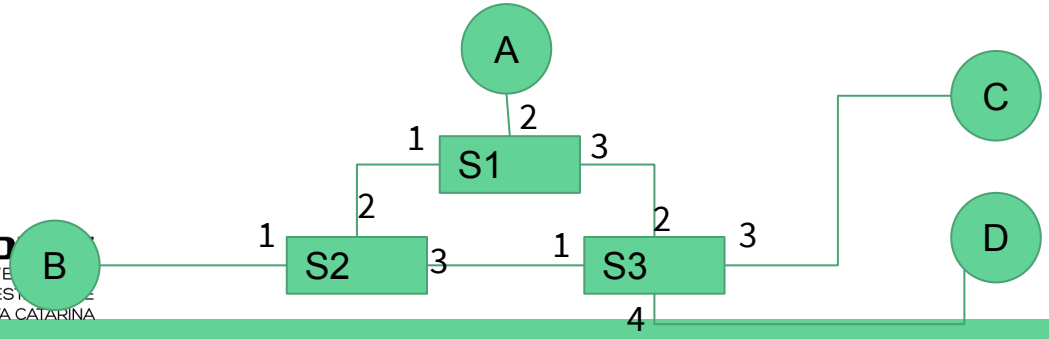
Exercícios

- Construa a seguinte topologia no mininet
  - Faça testes automatizados para verificar as larguras de banda entre os clientes
- Crie logs para os resultados
- Utilize atraso de 50ms em cada link



Exercícios

- Estabeleça manualmente (não precisa programar) as regras do *openflow* necessárias para que as seguintes políticas sejam viabilizadas na topologia abaixo:
  - Nós A, B, e C podem conversar entre si sem restrições
  - Nó D só pode acessar os nós A e B pelas portas 22 e 80
  - Nó D e C não podem se comunicar



Links úteis

- https://github.com/mininet/openflow-tutorial/wiki/Create-a-Learning-Switch
- http://www.brianlinkletter.com/using-the-pox-sdn-controller/
- https://openflow.stanford.edu/display/ONL/POX+Wiki