

Roteiro Lab4 - SDN

Parte 1 – Instalação e Configuração

1. Baixe o VirtualBox. Ferramenta para gerencia de maquina virtual. Em [virtualbox.org](https://www.virtualbox.org) baixe o pacote para seu SO

No Ubuntu (o meu é Ubuntu 20.04.3 LTS):

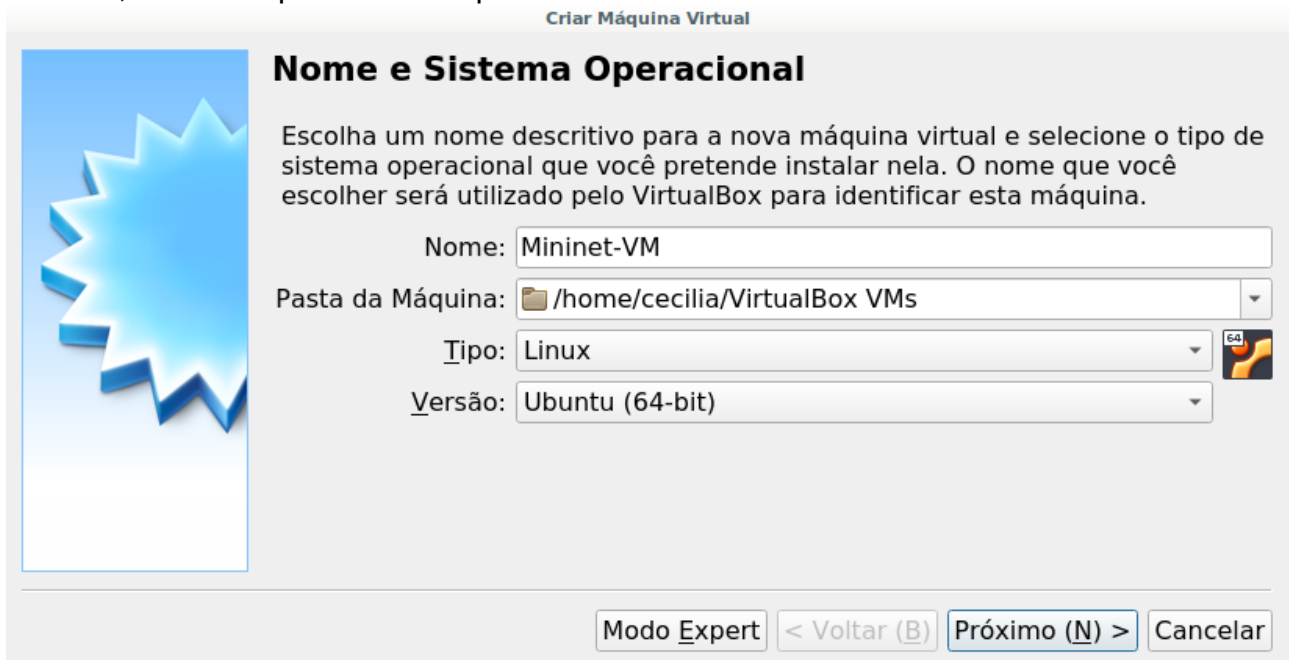
```
$ sudo dpkg -i virtualbox-6.1_6.1.26-145957_Ubuntu_eoan_amd64.deb
```

2. Baixe uma máquina virtual do mininet (é pesada! 954MB o zip)

Em <https://github.com/mininet/mininet/releases/> escolhi a 20.04.1

O arquivo está no formato OVF (Open Virtualization Format) que pode ser importada no virtualbox.

Crie uma nova VM no Vbox, Mininet-VM (sugestão 1G RAM), usando um disco rígido virtual existente, do tipo VMDK (que extrai do zip que acabei de baixar), dinamicamente alocado, de 8 GB para esta máquina => Criar.



Criar Máquina Virtual

Nome e Sistema Operacional

Escolha um nome descritivo para a nova máquina virtual e selecione o tipo de sistema operacional que você pretende instalar nela. O nome que você escolher será utilizado pelo VirtualBox para identificar esta máquina.

Nome:

Pasta da Máquina:

Tipo:

Versão:

Modo Expert < Voltar (B) Próximo (N) > Cancelar

Criar Máquina Virtual



Disco rígido

Se você quiser, pode acrescentar um disco rígido virtual a esta máquina virtual. Você pode acrescentar um arquivo de disco rígido virtual na lista ou selecionar outro local utilizando o ícone de pasta.

Se você deseja uma configuração de armazenamento mais complexa, pode pular este passo e fazer as mudanças manualmente na configuração da máquina assim que ela terminar de ser criada.

Recomenda-se utilizar um disco rígido de **10,00 GB**.

☐ Não acrescentar um disco rígido virtual
☐ Criar um novo disco rígido virtual agora
☒ Utilizar um disco rígido virtual existente

mininet-vm-x86_64.vmdk (Normal, 8,00 GB)

< Voltar (B)
Criar
Cancelar

Ao iniciar a VM já deveria conseguir pingar a internet.

Todavia é necessário que a sua máquina virtual também enxergue a máquina hospedeira. Então na configuração de rede, além de ter uma saída via NAT para a internet, criar uma saída do tipo host-only para formar uma rede local com a máquina hospedeira.

Para criar esta interface host-only:

Em Arquivo => HostNetworkManager => Criar, se cria uma interface de rede chamada vboxnet0 com DHCP habilitado.

Gerenciador de Redes do Hospedeiro

Rede (N)

Criar
 Remover
 Propriedades

Nome	Endereço IPv4/Máscara	Endereço IPv6/Máscara	Servidor DHCP
vboxnet0	192.168.56.1/24		<input checked="" type="checkbox"/> Habilitar

⌵ Fechar

A seguir nas configurações da VM, habilitar outro adaptador de rede virtual.

Configurações → Rede → Adaptador 2. Vai aparecer o nome vboxnet0 que foi criado no passo anterior.

Mininet-VM - Configurações

- Geral
- Sistema
- Monitor
- Armazenamento
- Áudio
- Rede
- Portas Seriais
- USB
- Pastas Compartilhadas

Rede

Adaptador 1
Adaptador 2
Adaptador 3
Adaptador 4

☒ Habilitar Placa de Rede

Conectado a: Placa de rede exclusiva de hospedeiro (host-only)

Nome: vboxnet0

▶ Avançado (D)

Iniciar a VM (login: mininet, password: mininet)

Na minha máquina hospedeira, o ifconfig deu:

```
(base) cecilia@Cecilia-Razer:~$ ifconfig
enx803f5d0b61f8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.15.8 netmask 255.255.255.0 broadcast 192.168.15.255
    inet6 fe80::bf9c:5593:da15:c875 prefixlen 64 scopeid 0x20<link>
    inet6 2804:431:e7ce:7ab:e300:8bdd:572b:bda prefixlen 64 scopeid 0x0<global>
    inet6 2804:431:e7ce:7ab:6b4a:6422:77c2:8be6 prefixlen 64 scopeid 0x0<global>
    ether 80:3f:5d:0b:61:f8 txqueuelen 1000 (Ethernet)
    RX packets 1064628 bytes 1391773564 (1.3 GB)
    RX errors 0 dropped 190 overruns 0 frame 0
    TX packets 277657 bytes 40411160 (40.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Loopback Local)
    RX packets 9373 bytes 915230 (915.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 9373 bytes 915230 (915.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vboxnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::800:27ff:fe00:0 prefixlen 64 scopeid 0x20<link>
    ether 0a:00:27:00:00:00 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 72 bytes 11156 (11.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Definir um IP fixo para a interface host-only da VM. No Linux:

\$ifconfig eth1 192.168.56.100

Na VM o ifconfig deu:

```
mininet@mininet-vm:~$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    ether 08:00:27:a1:ae:50 txqueuelen 1000 (Ethernet)
    RX packets 504 bytes 44691 (44.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 506 bytes 43887 (43.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.100 netmask 255.255.255.0 broadcast 192.168.56.255
    ether 08:00:27:ce:b3:d7 txqueuelen 1000 (Ethernet)
    RX packets 6 bytes 512 (512.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 512 (512.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 30 bytes 2950 (2.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30 bytes 2950 (2.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Verificar se todo mundo pinga todo mundo (internet e maquina local partindo de qualquer lado)

A VM do mininet não vem com uma interface gráfica, então é necessário jogar a saída gráfica para fora.

- No Windows:
 - Rodar o *XMing* como admin no Windows. Este programa vai aceitar a saída gráfica no Windows.
 - Faça um *ssh* da hospedeira para o Mininet (endereço fixado acima): No programa *putty*, coloque o IP do HostOnly e marque na opção *X11 -> forwarding* como Enable , isto quer dizer que a saída x11 será enviada para o Xming. Faça novamente o login como usuário mininet, senha mininet.
 - Na janela aberta pelo ssh, chame o emulador de terminal em background: *xterm&*. Pode ativar várias janelas simultaneamente.
- No Linux:
 - Avise a VM que a saída gráfica do X11 deve ser enviada para a janela do *ssh*, onde você se loga como usuário mininet:
ssh -X mininet@192.168.56.100

Teste a ativação do mininet:

```
$ sudo mn --topo single,3 --mac --controller remote --switch ovsk, protocols=OpenFlow13
```

Este comando cria uma topologia simples com um switch e 3 nós ligados a ele, com mac.... E um controlador na porta default 6633. Utiliza um switch virtual

Alguns comandos para investigar a topologia criada:

```
> nodes
> ports
> links
> h1 ping h2
```

Sem um controlador programado, o ping deveria falhar.

Até aqui o mininet está usando um controlador default (o c0), agora vamos incorporar o Ryu

```
$git clone git://github.com/osrg/ryu.git
$cd ryu
$sudo python3 ./setup.py install
```

Instalei pacotes adicionais que faltavam no site do Ryu:
(antes de mais nada atualizar versão do pip):

```
$sudo apt-get update
$sudo apt-get install python3-pip
$pip install -r tools/optional-requirements
```

Ir resolvendo todas as reclamações (a seguir um recorte dos problemas em minha instalação).

```
$sudo pip3 install netaddr
```

```
mininet@mininet-vm:~/ryu$ sudo pip3 install netaddr
Collecting netaddr
  Downloading netaddr-0.8.0-py2.py3-none-any.whl (1.9 MB)
    |████████████████████████████████████████| 1.9 MB 8.1 MB/s
ERROR: ryu 4.34 requires eventlet==0.31.1, which is not installed.
ERROR: ryu 4.34 requires msgpack>=0.4.0, which is not installed.
ERROR: ryu 4.34 requires oslo.config>=2.5.0, which is not installed.
ERROR: ryu 4.34 requires packaging==20.9, which is not installed.
ERROR: ryu 4.34 requires routes, which is not installed.
ERROR: ryu 4.34 requires tinyrpc==1.0.4, which is not installed.
ERROR: ryu 4.34 requires webob>=1.2, which is not installed.
Installing collected packages: netaddr
Successfully installed netaddr-0.8.0
```

Em alguns casos, a versão mais nova era incompatível com o Ryu, então instalar versão específica, como o eventlet abaixo:

```
mininet@mininet-vm:~/ryu$ pip3 install eventlet==0.31.1
Collecting eventlet==0.31.1
  Using cached eventlet-0.31.1-py2.py3-none-any.whl (224 kB)
Collecting dnspython<2.0.0,>=1.15.0
  Using cached dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
Requirement already satisfied: six>=1.10.0 in /usr/lib/python3/dist-packages (from eventlet==0.31.1) (1.14.0)
Requirement already satisfied: greenlet>=0.3 in /usr/local/lib/python3.8/dist-packages (from eventlet==0.31.1) (1.1.2)
ERROR: ryu 4.34 has requirement packaging==20.9, but you'll have packaging 21.0 which is incompatible.
ERROR: ryu 4.34 has requirement tinyrpc==1.0.4, but you'll have tinyrpc 1.1.2 which is incompatible.
Installing collected packages: dnspython, eventlet
Successfully installed dnspython-1.16.0 eventlet-0.31.1
```

Ativar o Ryu:

```
mininet@mininet-vm:~/ryu$ sudo ./bin/ryu-manager --verbose ryu/app/simple_switch_13.py
loading app ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK SimpleSwitch13
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPSwitchFeatures
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'SimpleSwitch13': {'main'}}
  PROVIDES EventOFPSwitchFeatures TO {'SimpleSwitch13': {'config'}}
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPErrormsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7fc7ae296c70> address:('127.0.0.1', 48106)
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7fc7ae296700> address:('127.0.0.1', 48108)
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7fc7ae2966d0> address:('127.0.0.1', 48110)
```

`$ sudo mn --topo single,3 --mac --controller remote --switch ovsk, protocols=OpenFlow13`

Alguns comandos para investigar a topologia criada:

```
> nodes
> ports
> links
> h1 ping h2
```

Agora o ping deve funcionar.

Parte 2 –Suas aplicações

A parte mais trabalhosa do lab foi a parte 1. Compreendendo o simple-switch, é fácil alterá-lo. Criaremos 2 aplicações, sendo que cada aplicação vale 50% da nota.

1) Crie a aplicação “filhote” de firewall (ffw.py)

Você deve criar uma aplicação como o simple-switch que aprende os MAC e faz o encaminhamento, porém com uma filtragem – crie uma única regra de filtro no seu firewall: pacotes do protocolo ICMP destinados ao host 2 serão descartados.

Assim, o ping entre h1 e h3 deve funcionar mas entre qualquer host e h2 não deve funcionar.

2) Crie a aplicação Hub (hub.py).

Transforme seu simple-switch em um hub. O hub replica os pacotes em todas as suas portas menos na porta em que o pacote chegou. Na prática, todos os pacotes que são direcionados a um hi, chegarão a todos os outros hs.

Submeta no classroom:

- ffw.py
- hub.py
- Um relatório onde mostra os testes realizados que provam que suas aplicações funcionaram.