*If you are having trouble setting up the VM locally using Virtual Box, we provide one alternative: sign up for AWS and run your tests in an EC2 micro-instance. Learn more about it here.*

A step-by-step video is available:

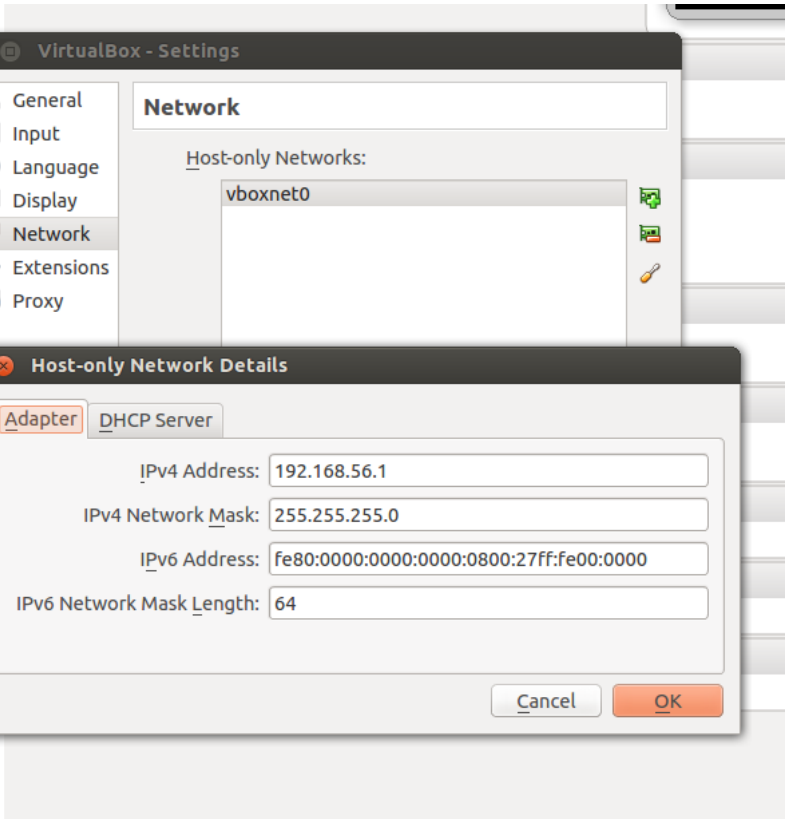
- on Youtube
- for download (~75Mo)

## Set up the VM with VirtualBox

- install VirtualBox, which is available for Windows, Mac and Linux
- download our VM image here. If your Virtual Box does not support the .ova format - because it's an older version-, you can download a legacy .ovf image here instead and extract the zip archive (which contains an .ovf file and the .vmdk disk image)
- import the .ova (or .ovf) file into VirtualBox using the Import Appliance menu
- start the VM, the credentials are: *mininet* / *mininet*

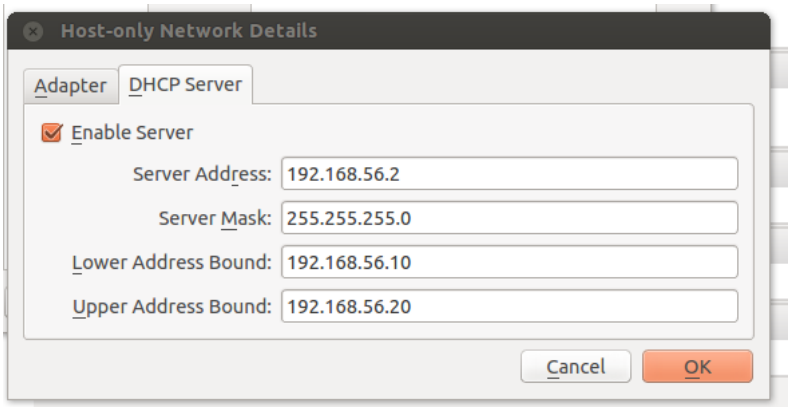
*If you are running Windows, it is probably best if you run the 32-bit image of the VM instead. You can download it here: 32-bit .ova. If you need the .ovf image, download it here: 32-bit .ovf.* More information can be found here.

To ssh into your VM -using the terminal handed to you by Virtual Box will not get you very far-, you will need to assign an 'Host-only adapter' to the VM. The first thing to do is create it in Virtual Box's preference menu, under Network. By default, the name for the adapter is vboxnet0. Check that the Adapter configuration looks like this:



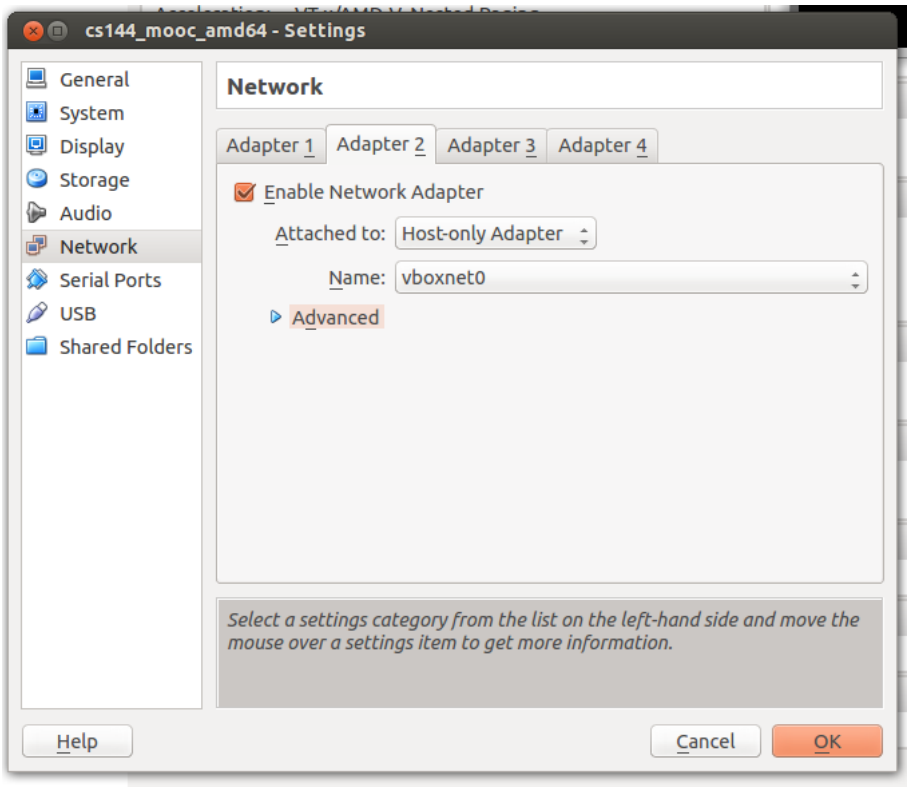

If the IPv4 address is different, you may want to change it to the same value as us to avoid possible conflicts later on.

You will then need to enable the DHCP server for this adapter. Use the following settings:




This will ensure that your VM receives an IPv4 address (in the range 192.168.56.10 - 192.168.56.20) when it starts. By default, this IP address will be the first one, 198.168.56.10, unless you have other VMs using the same adapter, which we recommend against.

The next step is to make sure that the adapter was properly assigned to the VM. Go to VM Settings -> Network, and make sure that the "Adpater 2" configuration looks like this:




You should now be able to ssh into the VM after powering it on (with the same credentials: *mininet* / *mininet*). On Linux and Mac, simply type:

```
ssh mininet@192.168.56.10
```

On Windows, you need to use third-party software (e.g. PuTTY).

If the above IP address does not work, you need to log into the VM using the Virtual Box terminal and find out the IP address assigned to the eth1 interface by typing:

```
ifconfig eth1
```

The address should be in the range 192.168.56.10 - 192.168.56.20. Otherwise, you did not configure the 'Host-only adapter properly').

## Run the testing environment

Power on the VM and ssh into it. The first thing to do is make sure that you have the most recent version of the starter code and reference solution binary:

```
bash# cd cs144_lab3
git pull --rebase
```

The assignment relies on two tools: Mininet and POX. Mininet emulates a network with a single router and POX ensures that this router can communicate with your code. To make your job easier, we have written scripts that start up Mininet and POX in the proper order. You simply need to run:

```
bash# ./run_all.sh
```

The script starts Mininet and POX in 2 different [screen](http://en.wikipedia.org/wiki/GNU_Screen) sessions. You can check that both are running correctly by attaching to each one.

For Mininet: Attach to the screen using 'screen -r mn'. You should see something like this:

```
*** Shutting down stale SimpleHTTPServers
*** Shutting down stale webservers
server1 192.168.2.2
server2 172.64.3.10
client 10.0.1.100
sw0-eth1 192.168.2.1
sw0-eth2 172.64.3.1
sw0-eth3 10.0.1.1
*** Successfully loaded ip settings for hosts
 {'server1': '192.168.2.2', 'sw0-eth3': '10.0.1.1', 'sw0-eth1': '192.
*** Creating network
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
client server1 server2
*** Adding switches:
sw0
*** Adding links:
(client, sw0) (server1, sw0) (server2, sw0)
*** Configuring hosts
client server1 server2
*** Starting controller
*** Starting 1 switches
sw0
*** setting default gateway of host server1
server1 192.168.2.1
*** setting default gateway of host server2
server2 172.64.3.1
*** setting default gateway of client client
client 10.0.1.1
192.168.2.0
172.64.3.0
10.0.1.0
*** Starting SimpleHTTPServer on host server1
*** Starting SimpleHTTPServer on host server2
*** Starting CLI:
mininet>
```

Detach the screen and return to the original terminal by typing Ctrl-a Ctrl-d (not just Ctrl-d !!!).

For POX: 'screen -r pox' should show something like this:

```
POX 0.0.0 / Copyright 2011 James McCauley
DEBUG:.home.mininet.cs144_lab3.pox_module.cs144.ofhandler:*** ofhandler: Successfully loaded ip sett
 {'server1': '192.168.2.2', 'sw0-eth3': '10.0.1.1', 'sw0-eth1': '192.168.2.1', 'sw0-eth2': '172.64.3

INFO:.home.mininet.cs144_lab3.pox_module.cs144.srhandler:created server
DEBUG:.home.mininet.cs144_lab3.pox_module.cs144.srhandler:SRServerListener listening on 8888
DEBUG:core:POX 0.0.0 going up...
DEBUG:core:Running on CPython (2.7.4/Apr 19 2013 18:28:01)
INFO:core:POX 0.0.0 is up.
This program comes with ABSOLUTELY NO WARRANTY.  This program is free software,
and you are welcome to redistribute it under certain conditions.
Type 'help(pox.license)' for details.
DEBUG:openflow.of_01:Listening for connections on 0.0.0.0:6633
Ready.
POX> INFO:openflow.of_01:[Con 1/108531946806346] Connected to 62-b5-90-22-dc-4a
DEBUG:.home.mininet.cs144_lab3.pox_module.cs144.ofhandler:Connection [Con 1/108531946806346]
DEBUG:.home.mininet.cs144_lab3.pox_module.cs144.srhandler:SRServerListener catch RouterInfo even, in
('172.64.3.1', '16:bc:76:50:2d:3f', '10Gbps', 2), 'eth1': ('192.168.2.1', '9a:5f:a0:7c:a1:65', '10Gb
th3'), ('192.168.2.2', '192.168.2.2', '255.255.255.255', 'eth1'), ('172.64.3.10', '172.64.3.10', '25
```

Once again, detach the screen with Ctrl-a Ctrl-d.

Now all that is left to do is run the router logic (i.e. the code you need to write). To check that the setup was done properly, you should start by running the reference solution binary that we provide (~/cs144_lab3/sr_solution on the VM). You can either run it in the current ssh terminal (which is not a good idea because then you will not have a prompt anymore and will not be able to run any tests), in a new ssh terminal (simply open a new connection) or in a screen session. Let's see how to do it with screen:

```
bash# screen -S sr
cd cs144_lab3/
./sr_solution
```

The output should look like this:

```
mininet@mininet-vm:~/cs144_lab3$ ./sr_solution
Using VNS sr stub code revised 2009-10-14 (rev 0.20)
Loading routing table from server, clear local routing table.
Loading routing table
-----------------------------------------
Destination     Gateway         Mask    Iface
0.0.0.0         10.0.1.100      0.0.0.0 eth3
192.168.2.2             192.168.2.2     255.255.255.255 eth1
172.64.3.10             172.64.3.10     255.255.255.255 eth2
-----------------------------------------
Client mininet connecting to Server localhost:8888
Requesting topology 0
successfully authenticated as mininet
Loading routing table from server, clear local routing table.
Loading routing table
-----------------------------------------
Destination     Gateway         Mask    Iface
0.0.0.0         10.0.1.100      0.0.0.0 eth3
192.168.2.2             192.168.2.2     255.255.255.255 eth1
172.64.3.10             172.64.3.10     255.255.255.255 eth2
-----------------------------------------
Router interfaces:
eth3    HWaddr32:08:c6:65:32:42
        inet addr 10.0.1.1
eth2    HWaddr9e:cc:66:f6:d0:b5
        inet addr 172.64.3.1
eth1    HWaddrae:e8:be:ee:7f:ee
        inet addr 192.168.2.1
 <-- Ready to process packets -->
```

To detach the screen and return to the main terminal, use Ctrl-a Ctrl-d. You can return to the session anytime with 'screen -r sr'. If you want to kill the session while attached to it, use Ctrl-d.

If everything is running correctly, you should be able to run the following command from your ssh terminal:

```
bash# ping 192.168.2.2
```

To test your code, just use your own binary instead of the solution.

## How to run tests?

The Mininet topology is connected to the VM (through iface client-eth0). Therefore, it is possible to execute pings, traceroutes to server1 and server2 from the VM itself. Another good way of running tests is from the Mininet CLI. To get accesss to it, just attach the Mininet screen session (screen -r mn), then type the command you want to execute preceded by the name of the Mininet host (client, server1, server2) you want executing it. For example to ping host server1 from host server2, run:

```
mininet> server2 ping -c 3 server1
```

To traceroute the router's sw0-eth3 interface from server1, run:

```
mininet> server1 traceroute -n 10.0.1.1
```