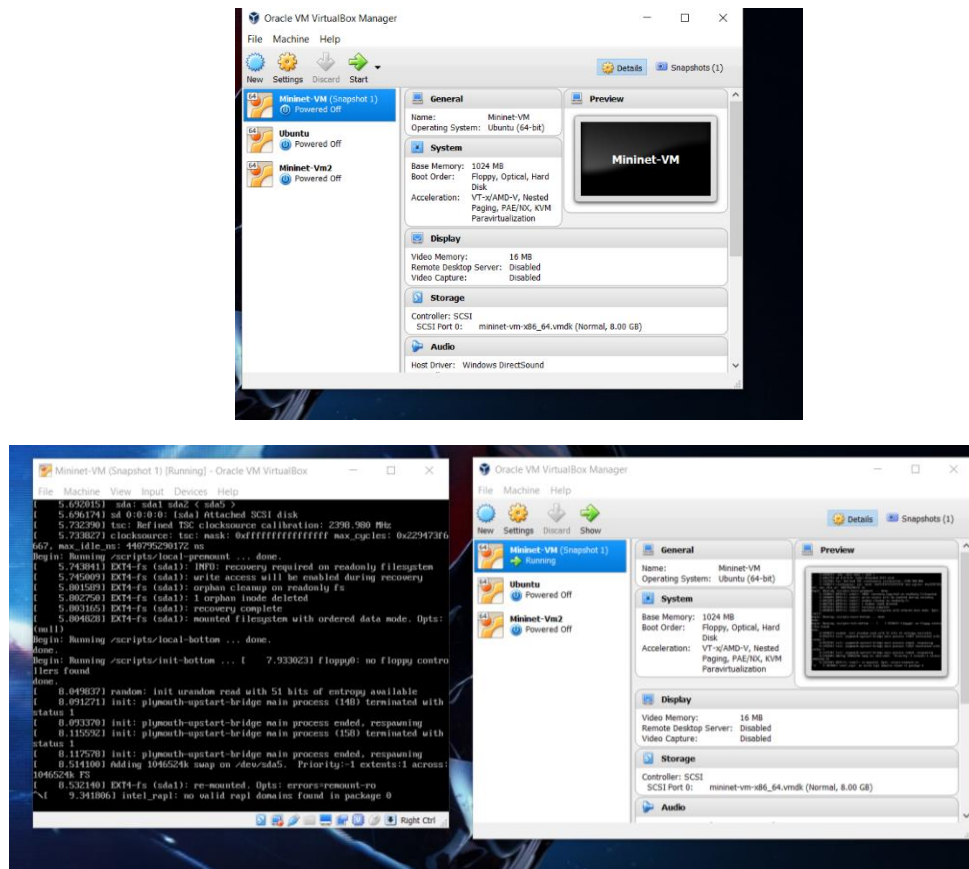


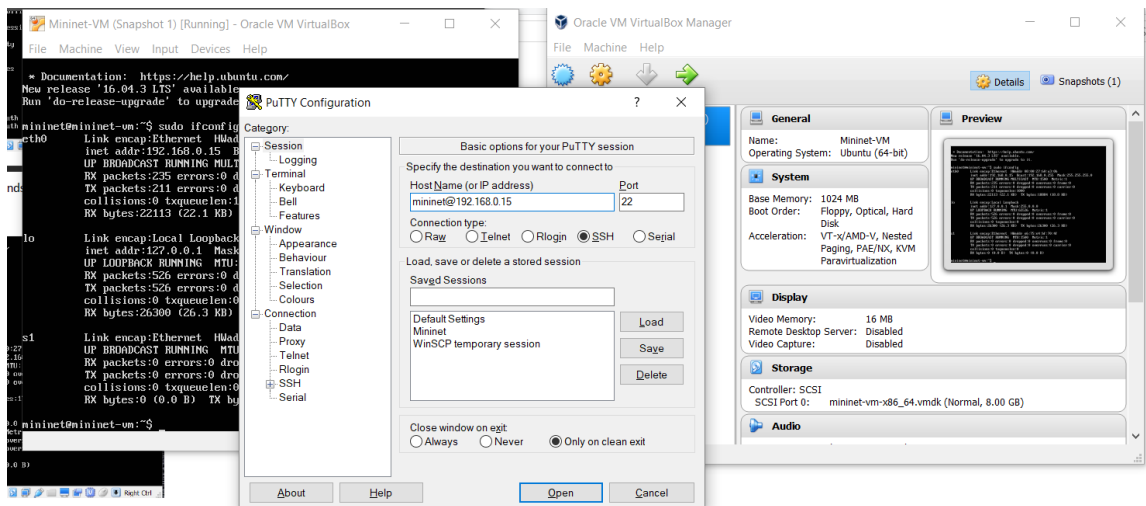
README

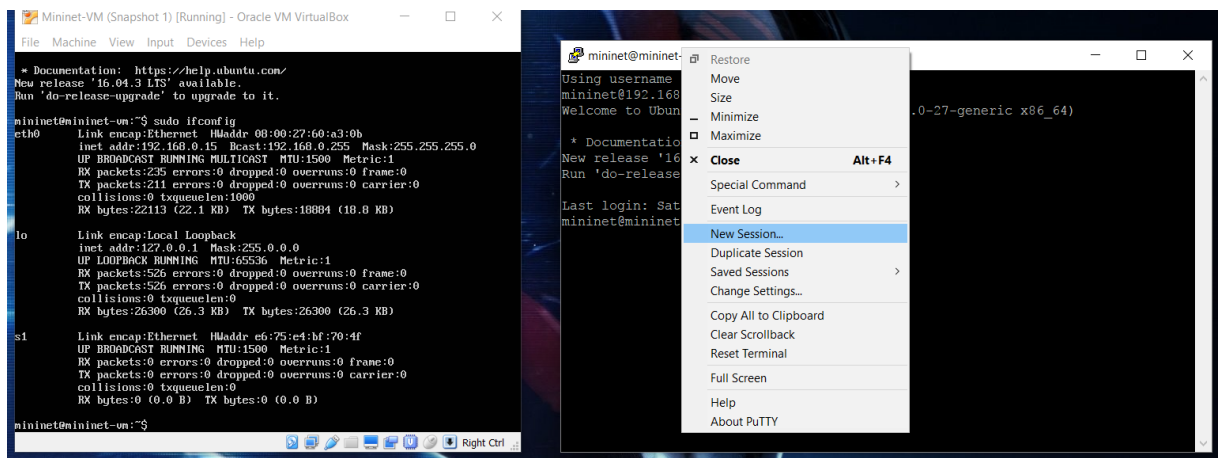
Instructions to run Advanced Load Balancer:

1. Run Virtual Machine in VirtualBox

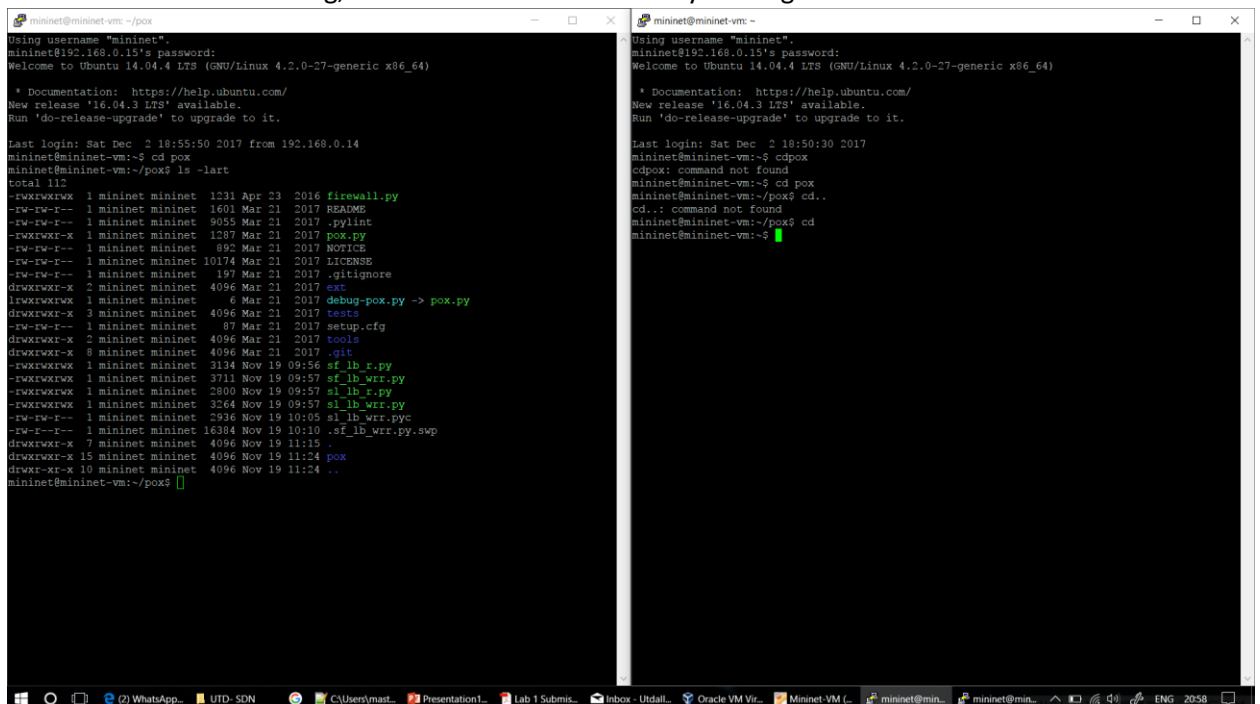


2. Run putty and login to mininet with the ip address of Virtual Machine and connection type SSH





3. Once the mininet is running, create a new session of Putty and login with mininet credentials



4. On one mininet instance, we run the POX controller and on the other mininet instance we run the topology-Switches, Hosts, Links
5. Copy the Python (Load Balancer) scripts to POX controller mininet session
6. On POX controller, run the Command: `sudo ./pox.py log.level --DEBUG sl_lb_wrr forwarding.I2_learning` .This will start the LoadBalancer (Stateless, WeightedRoundRobin).

```

mininet@mininet-vm: ~/pox
Using username "mininet".
mininet@192.168.0.15's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Dec 2 18:55:50 2017 from 192.168.0.14
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ ls -lart
total 112
-rwxrwxrwx 1 mininet mininet 1231 Apr 23 2016 firewall.py
-rw-rw-r-- 1 mininet mininet 1601 Mar 21 2017 README
-rw-rw-r-- 1 mininet mininet 9055 Mar 21 2017 .pylint
-rwxrwxr-x 1 mininet mininet 1287 Mar 21 2017 pox.py
-rw-rw-r-- 1 mininet mininet 892 Mar 21 2017 NOTICE
-rw-rw-r-- 1 mininet mininet 10174 Mar 21 2017 LICENSE
-rw-rw-r-- 1 mininet mininet 197 Mar 21 2017 .gitignore
drwxrwxr-x 2 mininet mininet 4096 Mar 21 2017 ext
lrwxrwxrwx 1 mininet mininet 6 Mar 21 2017 debug-pox.py -> pox.py
drwxrwxr-x 3 mininet mininet 4096 Mar 21 2017 tests
-rw-rw-r-- 1 mininet mininet 87 Mar 21 2017 setup.cfg
drwxrwxr-x 2 mininet mininet 4096 Mar 21 2017 tools
drwxrwxr-x 8 mininet mininet 4096 Mar 21 2017 .git
-rwxrwxrwx 1 mininet mininet 3134 Nov 19 09:56 sf_lb_r.py
-rwxrwxrwx 1 mininet mininet 3711 Nov 19 09:57 sf_lb_wrr.py
-rwxrwxrwx 1 mininet mininet 2800 Nov 19 09:57 sl_lb_r.py
-rwxrwxrwx 1 mininet mininet 3264 Nov 19 09:57 sl_lb_wrr.py
-rw-rw-r-- 1 mininet mininet 2936 Nov 19 10:05 sl_lb_wrr.pyc
-rw-rw-r-- 1 mininet mininet 16384 Nov 19 10:10 .sf_lb_wrr.py.swp
drwxrwxr-x 7 mininet mininet 4096 Nov 19 11:15 .
drwxrwxr-x 15 mininet mininet 4096 Nov 19 11:24 pox
drwxr-xr-x 10 mininet mininet 4096 Nov 19 11:24 ..
mininet@mininet-vm:~/pox$ sudo ./pox.py log.level --DEBUG sl_lb_wrr forwarding.l2_learning
pox 0.2.0 (carp) /usr/bin/python2.7.6 [2017-12-02] James McCauley, et al.
INFO:sl_lb_wrr:Stateless LB running...
DEBUG:core:Pox 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Oct 26 2016 20:30:19)
DEBUG:core:Platform is Linux-4.2.0-27-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:Pox 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-01 1]

```

- On the other mininet session, we create the topology using the following command:
`$ sudo mn --arp --topo single,4 --mac --switch ovsk --controller remote .`

```

mininet@mininet-vm: ~
Using username "mininet".
mininet@192.168.0.15's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Dec 2 18:50:30 2017
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ cd ..
mininet@mininet-vm:~$ sudo mn --arp --topo single,4 --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
Unable to connect the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>

```

- Now we select a virtual IP (VIP) and MAC for the load-balancer. This is the IP address to which the clients will make a HTTP request. The controller will push rules to rewrite the VIP with the selected HTTP server. To make this work, you need to static set an ARP entry for the VIP in the client. If 'h1' is the client and 10.0.0.5 is the VIP, the following command will add the static ARP entry: `h1 arp -s 10.0.0.5 00:00:00:00:00:05`

```
mininet@mininet-vm: ~$ sudo mn --arp --topo single,4 --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 arp -s 10.0.0.5 00:00:00:00:00:05
mininet>
```

9. Now traffic is generated by running SimpleHTTPServer webserver on the hosts by using following command.

- a. h1 python -m SimpleHTTPServer 80 &
- b. h2 python -m SimpleHTTPServer 90 &
- c. h3 python -m SimpleHTTPServer 100 &
- d. h4 python -m SimpleHTTPServer 110 &

```
mininet@mininet-vm: ~$ sudo mn --arp --topo single,4 --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 arp -s 10.0.0.5 00:00:00:00:00:05
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 python -m SimpleHTTPServer 90 &
mininet> h3 python -m SimpleHTTPServer 100 &
mininet> h4 python -m SimpleHTTPServer 110 &
mininet>
```

10. We can test pingability using the command: *pingall*

The screenshot shows a Mininet VM terminal with two windows. The left window displays a log of network events, including file transfers and connection attempts. The right window shows the output of a 'wget' command, which has successfully downloaded a directory listing from a server at 10.0.0.5:8000. The directory listing includes files like 'bash_history', 'bash_logout', 'bashrc', 'cache', 'gitconfig', 'mininet_history', 'profile', 'rnd', 'viminfo', 'vireshtark', 'xflops', 'oftest', 'openflow', and 'pox'. A ping command is also shown, indicating that the connection is successful.

11. Now when a client performs the following command, we will receive the IP address of the handling server. Command: `h1 wget http://10.0.0.5:8000`

Techniques used:

- Stateless Weighted Round Robin:** All requests coming from the client (h1) will be assigned to different servers (h2, h3, h4) on the order of their weights. Order of weights assigned: h2>h3>h4. So, the first request will be assigned to h2 followed by h3 and h4
- Stateless Random:** All requests coming from the client (h1) will be assigned to different servers (h2, h3, h4) on a random order.
- Stateful Weighted Round robin:** All requests coming from the client (h1) will be assigned to different servers (h2, h3, h4) on a random order. Order of weights assigned: h2>h3>h4. So, the first request will be assigned to h2 followed by h3 and h4. Also, as it is stateful so all requests from the same client will be directed to the same server.
- Stateful Random:** All requests coming from the client (h1) will be assigned to different servers (h2, h3, h4) on a random order. But all the requests from the same client will be directed to the same server as it is stateful.

12. Repeat the steps 6-11 for the other Python scripts to Implement different load balancing techniques.