



# Software Defined Load Balancer with POX Controller

**CS 6301.502 | Fall 2018**

Priyank Shah

Manan Lakhani

Karan Motani

Ankita Patil

Kripanshu Bhargava

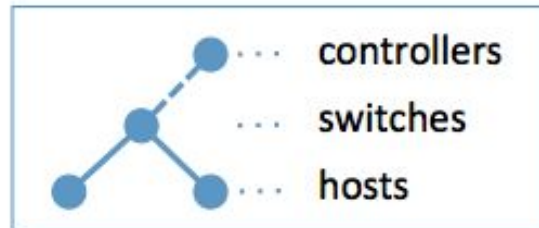
# Objective

To develop a software defined load balancer with POX controller which makes flow balancing decisions based on the statistics such as bandwidth and latency, gathered from interfaces of switches and nodes in the network



# Technology

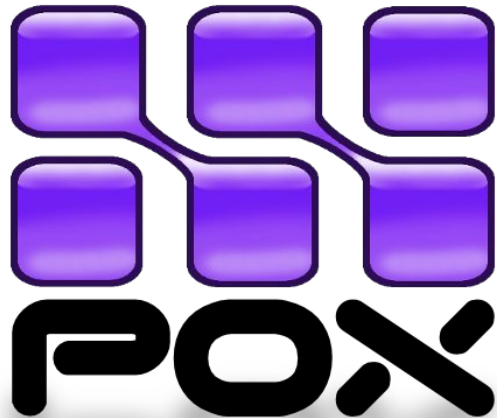
```
> sudo mn
```



## 1. Mininet

To simulate the network

# Technology



## 2. POX Controller

Open source development platform for Python based software defined SDN control application to check the configuration of customized networking application

# Technology



## **3. OpenDayLight**

To visually represent and simulate the network topologies

The background of the slide features several thin, curved lines in shades of gray, some solid and some dashed, creating a modern, abstract design.

# Network Topology

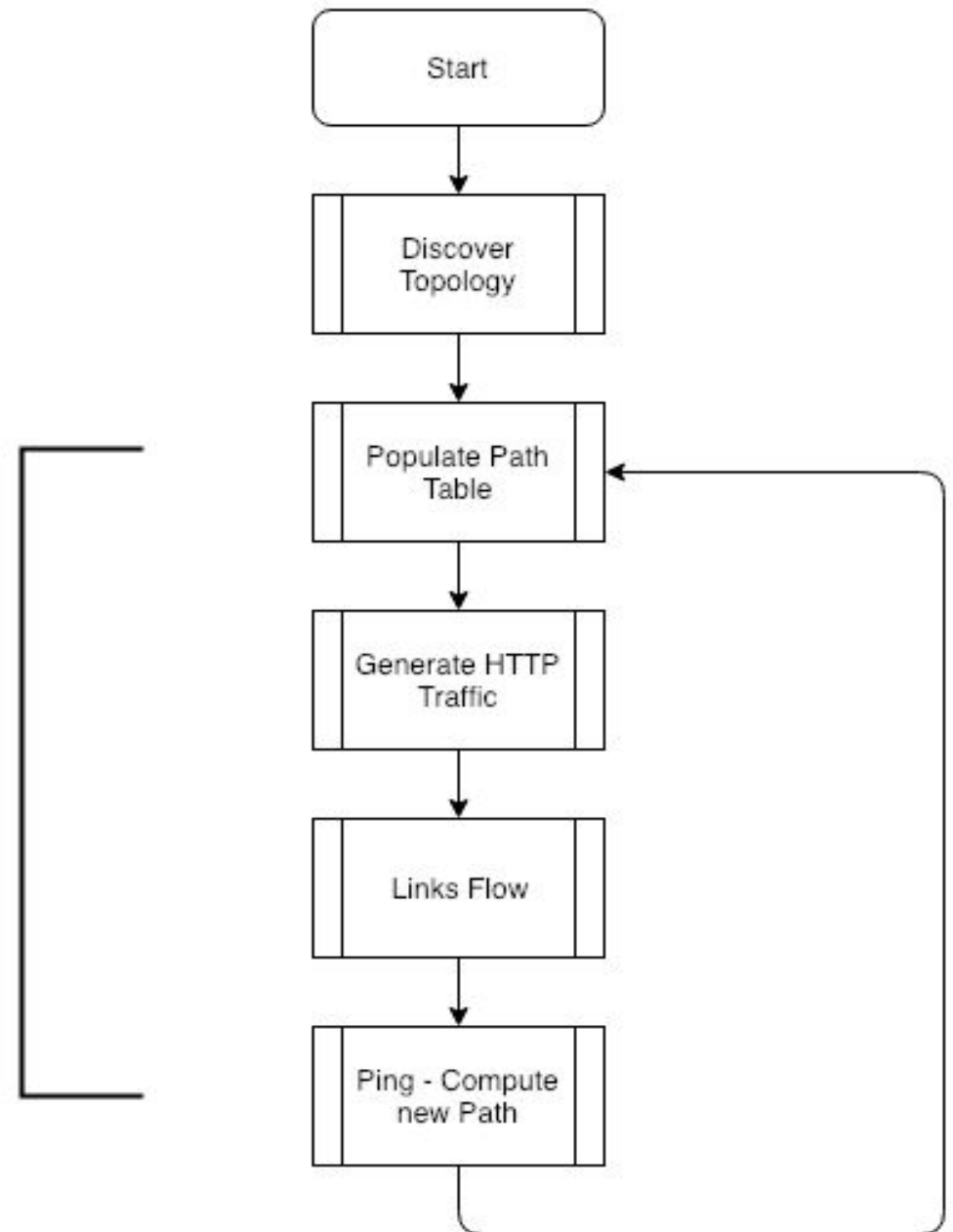
In mininet, a network topology is created which consists of three switches and three hosts. The hosts are labelled as h1, h2, h3 and switches are labelled as S1, S2, S3.

# Application Flow

## loop

This loop runs every 5 minutes in order to populate new tables for new links. loop is triggered using ping command from python file on mininet

### Load Balancer Application Flow





Screenshots



# POX Controller Setup

```
mininet@mininet-vm: ~/pox (ssh)
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG controller openflow.discovery openflow.of_01
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:controller:Reading paths file
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
```

# Creating Custom Topology (RING)

```
mininet@mininet-vm: ~/mininet/code (ssh)
mininet@mininet-vm:~/mininet/code$ sudo mn --custom ringTopo.py --topo mytopo --controller remote --mac --switch ovsk
*** 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
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s1, s2) (s2, s3) (s3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>

mininet@mininet-vm: ~/pox (ssh)
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG controller openflow.discovery openflow.of_01
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:controller:Reading paths file
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-Ubun
tu-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:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-03
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-01
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-02
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.3 ->
00-00-00-00-00-02.3
DEBUG:controller:Adding switch to database: 00-00-00-00-00-03
DEBUG:controller:Adding switch to database: 00-00-00-00-00-02
DEBUG:controller:link between 00-00-00-00-00-03:3 to 00-00-00-
00-00-02:3
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.2 ->
00-00-00-00-00-01.3
DEBUG:controller:Adding switch to database: 00-00-00-00-00-01
DEBUG:controller:link between 00-00-00-00-00-03:2 to 00-00-00-
00-00-01:3
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.3 ->
00-00-00-00-00-03.2
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.2 ->
00-00-00-00-00-02.2
DEBUG:controller:link between 00-00-00-00-00-01:2 to 00-00-00-
00-00-02:2
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.3 ->
00-00-00-00-00-03.3
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 ->
00-00-00-00-00-01.2
[]
```

# Starting HTTP Server

```
mininet@mininet-vm: ~/mininet/code (ssh)
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 wget -O - h1
--2018-12-03 14:21:28-- http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 406 [text/html]
Saving to: 'STDOUT'

 0% [                  ] 0          ---K/s          <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="ping1.py">ping1.py</a>
<li><a href="ping2.py">ping2.py</a>
<li><a href="ping3.py">ping3.py</a>
<li><a href="ringTopo.py">ringTopo.py</a>
<li><a href="starTopo.py">starTopo.py</a>
<li><a href="stats.py">stats.py</a>
</ul>
<hr>
</body>
</html>
100%[=====>] 406          ---K/s    in 0s

2018-12-03 14:21:28 (62.2 MB/s) - written to stdout [406/406]

mininet> █
```

# Generating Traffic and Collecting Statistics

✕ mininet@mininet-vm: ~/mininet/code (ssh)

```
mininet> h1 python ping1.py
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.2 - - [03/Dec/2018 14:21:28] "GET / HTTP/1.1" 200 -
mininet> h2 python ping2.py
mininet> h3 python ping3.py
mininet> █
```

✕ mininet@mininet-vm: ~/mininet/code (ssh)

```
mininet> sh python stats.py
1:2:3
2:1:3
1:3:3
3:1:1
3:2:2
2:3:3
mininet> █
```



Statistics

```
mininet> sh python stats.py
```

```
1:2:3
```

```
2:1:3
```

```
1:3:3
```

```
3:1:1
```

```
3:2:2
```

```
2:3:3
```

```
mininet> █
```

To get the statistics from the switches, a set of python scripts are written which will ping all the nodes from a particular node. After pinging the nodes, average latency between each node is collected. Depending upon the average latency, an alternative path with lowest overall latency is selected. This statistics are used by controller to modify flows.

```
import urllib, os
```

```
# This script is used for pinging the client 2 and 3 from 1
```

- `os.system("ping -c 5 10.0.0.2 | tail -1 | awk '{print $4}' | cut -d '/' -f 2 >> ping_stat_1_2")`
- `os.system("ping -c 5 10.0.0.3 | tail -1 | awk '{print $4}' | cut -d '/' -f 2 >> ping_stat_1_3")`

# Challenges

01

While setting up  
the whole project  
from a closed  
network

02

While setting up  
OpenDayLight

03

While setting up  
POX Controller

# Conclusion

Implemented software defined POX controller which makes flow balancing decisions based on statistics such as bandwidth and latency gathered from the interfaces of nodes and switches in the network which is simulated in mininet.





# References

- <http://mininet.org/walkthrough/>
- <https://www.opendaylight.org/>
- <https://openflow.stanford.edu/display/ONL/POX+Wiki.html>
- <http://networkstatic.net/installing-mininet-openshift-openshift-vswitch/>
- <https://searchnetworking.techtarget.com/definition/Mininet>



Thank you!