This document describes how to simulate a network using mininet, load the POX controller for install a ruleset and push traffic in the network.

**Sections:**

**0) Files**

**1) Running POX**

**2) Start Mininet**

**3) Send traffic**

**4) How to calculate the controller time**

**5) How to run others networks**

**0) Files**

The code its located in:

[**https://svn.cs.stonybrook.edu/repos/middleboxsdn/advpro/net-tunnel**](https://svn.cs.stonybrook.edu/repos/middleboxsdn/advpro/net-tunnel)

Here is the description of some important files related to the project.

conx\_\* its for the network topology of the switches and MB, port number and mac address.

Hosts\_\* its for the ip address of the networks and middleboxes

Switches\_\* the switches table

tgMaps\_\* the mapping file needed by the traffic generator

Ifconfig\_\* not relevant for now

Netconfig\_\* not relevant for now

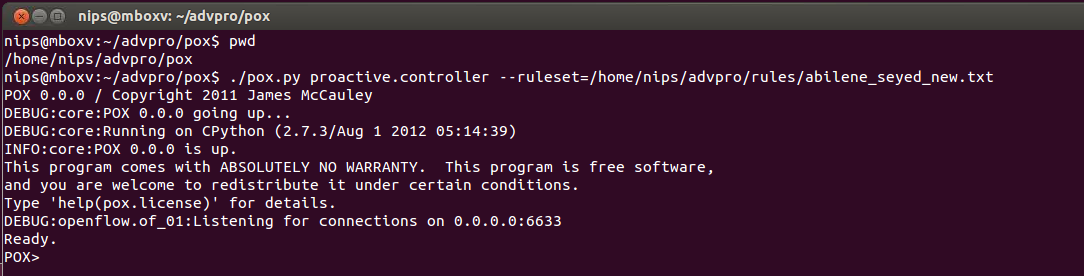
**1) Running POX**

If we want to install rules proactively we should have a ruleset ready before start the network and give it as a parameter to the POX controller.

To run the POX controller we need to go to the pox folder and execute it.

**./pox.py proactive.controller --ruleset=/path\_to\_ruleset/ruleset.txt**

You can change the parameter --ruleset to any other ruleset file. The ruleset its trying to follow the format of openvswitch but it could vary.



**2) Start Mininet**

Once we have running the POX controller we could start the network, by executing the python script **luis\_topo\_file.py** as sudo

sudo python luis\_topo\_file.py **<network\_name>** **<type\_of\_experiment>** **<path\_to\_topology\_files>**

Example:

sudo python luis\_topo\_file.py abileneSeyed new topology/Topology\_abileneSeyed

output:

nips@mboxv:~/advpro$ sudo python luis\_topo\_file.py abileneSeyed new topology/Topology\_abileneSeyed

\*\*\* Creating controllers

\*\*\* read topology from file

host ID: []

switch ID: ['S1', 'S2', 'S4', 'S3', 'S5', 'S6', 'S9', 'S8', 'S7', 'S11', 'S10']

MB ID: ['M1', 'M2', 'M3', 'M4', 'M5', 'M6', 'M7', 'M8', 'M9', 'M10', 'M11', 'M12']

Edge: [('S1', 'S2'), ('S1', 'S4'), ('S2', 'S3'), ('S2', 'S4'), ('S3', 'S5'), ('S4', 'S6'), ('S5', 'S6'), ('S5', 'S9'), ('S6', 'S8'), ('S7', 'S8'), ('S7', 'S11'), ('S8', 'S9'), ('S9', 'S10'), ('S10', 'S11'), ('S1', 'M1'), ('S2', 'M2'), ('S3', 'M3'), ('S4', 'M4'), ('S5', 'M5'), ('S6', 'M6'), ('S7', 'M7'), ('S8', 'M8'), ('S9', 'M9'), ('S10', 'M10'), ('S11', 'M11'), ('S9', 'M12')]

add switch:  S1

add switch:  S2

add switch:  S4

add switch:  S3

add switch:  S5

add switch:  S6

add switch:  S9

add switch:  S8

add switch:  S7

add switch:  S11

add switch:  S10

add traffic generator:  Tg

add middlebox:  M1

add middlebox:  M2

add middlebox:  M3

add middlebox:  M4

add middlebox:  M5

add middlebox:  M6

add middlebox:  M7

add middlebox:  M8

add middlebox:  M9

add middlebox:  M10

add middlebox:  M11

add middlebox:  M12

add link between two node :  S1 S2

add link between two node :  S1 S4

add link between two node :  S2 S3

add link between two node :  S2 S4

add link between two node :  S3 S5

add link between two node :  S4 S6

add link between two node :  S5 S6

add link between two node :  S5 S9

add link between two node :  S6 S8

add link between two node :  S7 S8

add link between two node :  S7 S11

add link between two node :  S8 S9

add link between two node :  S9 S10

add link between two node :  S10 S11

add link between two node :  S1 M1

add link between two node :  S2 M2

add link between two node :  S3 M3

add link between two node :  S4 M4

add link between two node :  S5 M5

add link between two node :  S6 M6

add link between two node :  S7 M7

add link between two node :  S8 M8

add link between two node :  S9 M9

add link between two node :  S10 M10

add link between two node :  S11 M11

add link between two node :  S9 M12

\*\*\* Starting network

\*\*\* Configuring hosts

Tg M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12

\*\*\* Sleep for reaching the controller (10 seconds)

\*\*\* View network

N1 10.1.0.1/16 Tg-eth0

N2 10.2.0.1/16 Tg-eth1

N3 10.3.0.1/16 Tg-eth2

N4 10.4.0.1/16 Tg-eth3

N5 10.5.0.1/16 Tg-eth4

N6 10.6.0.1/16 Tg-eth5

N7 10.7.0.1/16 Tg-eth6

N8 10.8.0.1/16 Tg-eth7

N9 10.9.0.1/16 Tg-eth8

N10 10.10.0.1/16 Tg-eth9

N11 10.11.0.1/16 Tg-eth10

M1 13.0.1.2 00:00:00:00:00:02 M1-eth0

M2 13.0.2.2 00:00:00:00:00:03 M2-eth0

M3 13.0.3.2 00:00:00:00:00:04 M3-eth0

M4 13.0.4.2 00:00:00:00:00:05 M4-eth0

M5 13.0.5.2 00:00:00:00:00:06 M5-eth0

M6 13.0.6.2 00:00:00:00:00:07 M6-eth0

M7 13.0.7.2 00:00:00:00:00:08 M7-eth0

M8 13.0.8.2 00:00:00:00:00:09 M8-eth0

M9 13.0.9.2 00:00:00:00:00:0a M9-eth0

M10 13.0.10.2 00:00:00:00:00:0b M10-eth0

M11 13.0.11.2 00:00:00:00:00:0c M11-eth0

M12 13.0.12.2 00:00:00:00:00:0d M12-eth0

\*\*\* configuring middleboxes

\*\*\* running iperf clients

mkdir: cannot create directory `whereiam': File exists

\*\*\* Starting CLI:

mininet>

**3) Send traffic**

Once we have the network up and running, the controller should have already installed the given rule set.

For verify the correctness of the ruleset we send traffic using a modified version of the **bittwistx** tool. That modification allow us to send traffic controlled by a single node simulating that its coming from multiple nodes in the network.

We need to open an xterm to the Traffic generator node (Tg)

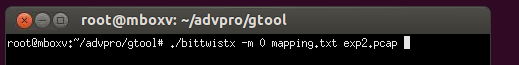
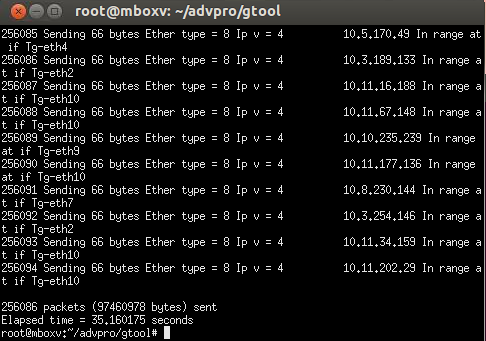
mininet> xterm Tg

then we execute the bittwistx in the new console giving the mapping file that indicates thought which interface we want to send traffic and the pcap file.

/net-tunnel/gtool/bittwistx -m 0 mapping.txt traffic.pcap

The **-m 0** its used to send all packages one after another, not considering the timing in the pcap file.

Example:

Then we end the experiment on mininet pressing Ctrl+D.  And the pcap file of the middleboxes are in the directory:

net-tunnel/pcap/

For show the flows traversing the switches we use at mininet:

mininet> dpctl dump-flows

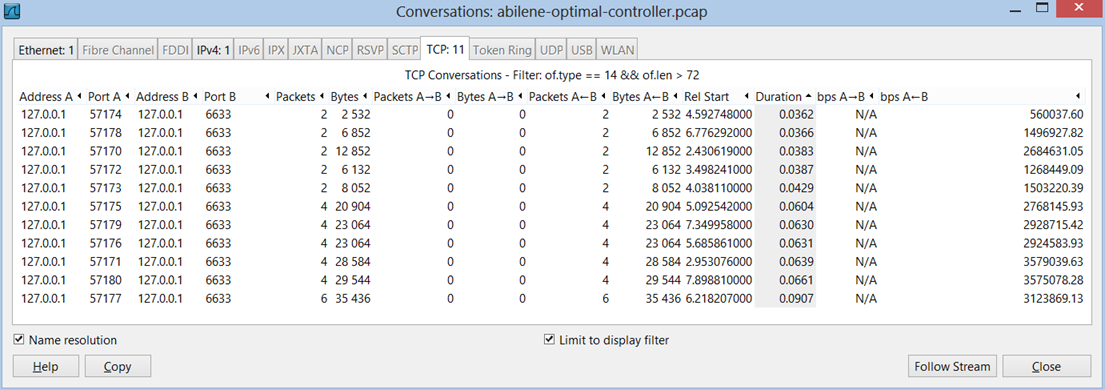
**4) How to calculate the controller time**

Start the experiment, including the POX controller, then check that POX install all the rules.

Then in the folder pcap/ will appear a file named \*-controller.pcap

Then open that file and apply the filter in wireshark: of.type == 14 && of.len > 72

Then go to Statistics -> Conversation, and enable “Limit to display filter”

then go to the tap TCP and should show the image bellow, pick the time of the longest TCP conversation. 

**5) How to run others networks**

sudo python luis\_topo\_file.py NetworkName RuleSetType topology/TopologyFile

NetworkName = The name of the network you are testing, that variable is used to the output fields

RuleSetType = the type of rule set that is beign tested

TopologyFile = the topology file

./pox.py proactive.controller --ruleset=/home/nips/advpro/rules/abilene\_seyed\_new.txt

replace with the path to the ruleset you want to test

Then continue with step 3 above.