Prevent L2 loops using Mininet and POX

Lennart Franked

December 6, 2024

Contents

1 Introduction

1.1 Aim

- Analyze the principles of SDN, including their architecture, protocols, and open-source solutions like OpenFlow.
- Apply SDN concepts in network automation.

1.2 Reading

Before starting this laboratory exercise, you should start by reading about Mininet [3] and get acquainted with *"Introduction to Mininet "* [1]
You will also need to get acquainted with POX, see *POX Documentation* [5]. You can download POX at [4].
Besides these sources, you might have to look directly in the source code for POX, to understand what paramaters certain classes and functions want. It might also be helpful to refer to the OpenFlow switch specifications [6].

2 About

In this laboratory excercise you will first install and work with Mininet, which will involve writing smaller Python scripts to build a network. Next you will install and connect an external POX controller to this network, and let it manage your switches. Finally you will build a new topology in mininet, connect your POX-controller to this network, and then write your own POX-controller that will be able to handle a special case, involving layer two loops.

2.1 Mininet

Mininet is network emulator that takes advantage of the namespaces in Linux to create a virtual network on your machine. This has the advantage that it uses very little resources, and it allows you to build fairly complex networks with minimal system resources. However Mininet use the same host file system and PID namespace, hence the virtual hosts added in mininet share a lot of resources. It supports OpenFlow which makes it a splendid candidate for running an SDN lab environment.

2.2 POX

POX is a SDN controller, written in Python, it is based on the NOX-controller. POX is event driven, and fairly low level. Hence when writing a POX-component, it is easier to not start from scratch, but instead take an existing component and modify that to our needs.

3 Installing and working with Mininet and POX

3.1 Installations

This laboratory assignment should be done in a virtual machine running a minimal Linux server, e.g Ubuntu-server. No guide for this will be given.

3.1.1 Mininet

Mininet is available in most Linux distributions package managers. As an example on how to install it in Ubuntu-server, see listing 1

Listing 1: Installing mininet

```
root@poxlab:~# apt install mininet
```

3.1.2 POX

POX is not available through the distributions package manager, instead you will need to download it from the projects GitHub. See [2] for detailed instructions on how to install POX.
Steps involved is the following:

- git clone http://github.com/noxrepo/pox

At time of writing, POX 0.7.0 (gar) does not support Python 3.10 and above. Hence ensure that you can run POX for example in `pyenv` under 3.9.
For a short guide on how to setup a Python environment, see appendix A.

To answer in your report For this task, answer the following questions in your report:

- Write a short summary about how you installed Mininet and POX. Your text must include any issues you had, and how you solved them.

3.2 Working with Mininet and POX

Before starting the actual assignment, it can be a good idea to get acquainted with Mininet and POX. Since you will most likely have to run Mininet multiple times during this assignment, it is strongly advised that you create your network with the help of a python script.

3.2.1 Building a network in Mininet

Start by following the steps in [1, Working with Mininet] and create a sample topology with three switches, where each switch has one host connected to it (three hosts in total).
In your Python script add the mininet.CLI so that you can interact with your mininet topology.
If you get an error regarding no default controller, try installing `openvswitch-testcontroller` and update your script accordingly:

```
from mininet.node import OVSController
...
def simpleTest():
    "Create and test a simple network"
    topo = SingleSwitchTopo(k=4)
    net = Mininet(topo, controller = OVSController)
...
```

Run your Python script and then when in Mininet

- ping host h1 to h3 using the command `h1 ping h3`.
- try adding a link between S1 and S3, so that you get a ring topology.
  - *You can turn on or off links between two nodes using the command* `link <node1> <node2> up|down`

In Mininet, notice how adding a link affects the connectivity between the hosts.

To answer in your report For this task, answer the following questions in your report:

- Include as an appendix to your report, the python script you wrote for this task. Your script must be well commented.
- Include a screenshots of your running mininet network, before and after you have added one extra link.
- A short reflection regarding the results of this task.

3.2.2 Adding a remote SDN controller

The default OVS controller, is designed to be simple, and cannot manage complex topologies, for example ring topologies. Therefore to handle a ring-type topology in Mininet, you need to replace the OVS controller with a more comprehensive controller, for example POX, that you downloaded in section 3.1.2. While keeping the original topology with three switches, and without redundant links. Add the POX controller to your Mininet script. In order to achieve this, you must first import `RemoteController` from `mininet.node`.
Then update your Mininet class accordingly:

```
net = Mininet(topo, controller=None)
net.addController ('c0',
                   controller=RemoteController,
                   ip='127.0.0.1',
                   port=6633 )
```

Finally, before you run your mininet-script, start your POX-controller in a separate terminal by running:

```
root@poxlab:~/pox# python pox.py
```

Try to ping h3 from h1 again. You will notice that you will not have any connectivity. This is because POX haven't been loaded with any components, and therefore do not know what to do with the frames it receives.
Stop the POX controller, and then restart it with the forwarding.l2_learning component.

```
root@poxlab:~/pox# python pox.py forwarding.l2_learning
```

Try to ping host h3 from h1. This time you should have connectivity.

In the Mininet CLI, add the link between S1 and S3 again. Once again notice how this affects the connectivity between the hosts.
The L2_learning component in POX, is only designed to make the switches work as regular switches, that is, it learns which MAC-address is connected to which port, and then forwards the frames accordingly. It doesn't handle ring topologies.
There is a POX component called "spanning_tree" that is made to handle these types of topologies. Therefore, remove the link between S1 and S3 and then restart your POX controller however this time loading the "sample.spanning_tree" component instead.

```
root@poxlab:~/pox# python pox.py samples.spanning_tree
```

Ensure that the connectivity between h1 and h3 is still working, then add the link between S1 and S3 again.
How did this affect the connectivity between the hosts?

To answer in your report For this task, answer the following questions in your report:

- Include as an appendix to your report, the python script you wrote for this task. Your script must be well commented.
- Include a screenshots of your running mininet network, showing that your network are using the POX controller.
- A short reflection regarding the results of this task.

4 Assignment

4.1 Building lab topology

You are now acquainted with Mininet and POX, and it is time to start with the laboratory task.

- Write a python script that creates a topology that looks like the one shown in fig. 1.
- Make sure to set your mininet IP-base to 10.YY.MM.0/24, where YY and MM is your year and month.
- Ensure that the IP-address for each host is set.

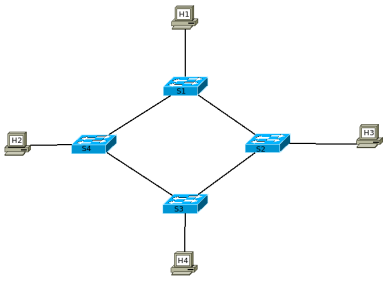

Figure 1: Base Topology for lab assignment

To answer in your report For this task, answer the following questions in your report:

- Include as an appendix to your report, the python script you wrote for this task. Your script must be well commented.

4.2 Configure POX

As shown in fig. 1, there are four switches connected in a ring-based topology. Your task will be to write your own POX component that can handle a ring topology. Your component should be based on the `l2_learning` component that you used in section 3.2.2.

1. Go to your POX-folder, and start by making a copy of `l2_learning`, that is located in `forwarding`, to the `ext` folder.
    - *The `ext` has been specifically created for people to add their own POX-components. It is therefore suitable to place your copy of `l2_learning` there.*
2. Rename the `ext/l2_learning.py` component to something that includes your student-id, for example `lefr0500_l2_loop_prevention.py`.

Since this is a small proof-of-concept laboratory assignment, your task will be to update the L2_learning component with the following features:

Task1: Your controller component should keep track of how your switches are connected with each other.
HINT: This is achieved with LLDP packets.

Task2: If your switch S3, is connected to both S2 and S4, it will close the port on S3 that is connected to S2. Hence breaking the loop.
HINT: This should be run on an event.

Task3: If the link between S3 and S4 goes down, your controller should enable the port on S3 that goes towards S2.
HINT: You will have to flush the flow tables on the switches.

Task4: Ensure that if the link between S3 and S4 comes back up, your controller should once again turn off the port on S3 that is connected to S2.

To answer in your report For this task, answer the following questions in your report:

- Include as an appendix to your report, the POX component you wrote for this task. Your script must be well commented.
- A detailed explanation of how your component works.

4.3 Verify that your component is working

Verify that your pox controller is working by:

- Start POX and load your component
- Start Mininet, ensure that it uses your POX-controller
- Start a ping between H4 and H3, does it work?
- Break the link between S3 and S4
- Does the connection still work?

To answer in your report For this task, answer the following questions in your report:

- Include screenshots that shows that your component works as intended.
- A short reflection regarding the results of this task.

5 Examination

Answer all the questions given in each subsection in section 4

6 References

References

[1] Brandon Heller "Bob Lantz Nikhil Handigol and Vimal Jeyakumar". *"Introduction to Mininet "*. 2021. URL: `https://github.com/mininet/mininet/wiki/Introduction-to-Mininet` (visited on 2024).

[2] "noxrepo". *"Installing POX"*. 2020. URL: `https://noxrepo.github.io/pox-doc/html/` (visited on 2024).

[3] Mininet". *"Mininet"*. 2022. URL: `https://mininet.org/` (visited on 2024).

[4] Noxrepo. *POX*. 2021. URL: `https://github.com/noxrepo/pox` (visited on 10/08/2024).

[5] Noxrepo. *POX Documentation*. 2015. URL: `https://noxrepo.github.io/pox-doc/html/` (visited on 10/08/2024).

[6] *OpenFlow Switch Specification*. Version 1.5.1. https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf. Open Networking Foundation. Mar. 2015.

A Setting up a Python environment

Setting up pyenv includes the following:

```
# Install dependencies for pyenv
root@poxlab:~# apt install -y make build-essential libssl-dev zlib1g-dev\
                libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm\
                libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev\
                libffi-dev liblzma-dev

# Install Pyenv
root@poxlab:~# curl https://pyenv.run | bash

# Add pyenv to PATH
root@poxlab:~# echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
root@poxlab:~# echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
root@poxlab:~# echo 'eval "$(pyenv init --path)"' >> ~/.bashrc
root@poxlab:~# echo 'eval "$(pyenv init -)"' >> ~/.bashrc

# Reload .bashrc
root@poxlab:~# source ~/.bashrc

# Check that pyenv works.
root@poxlab:~# pyenv --version
pyenv 2.4.11

#Install Python 3.9-7.3.16
root@poxlab:~# pyenv install pypy3.9-7.3.16

#Set Python 3.9-7.3.16 as the local version in the POX directory
root@poxlab:~# cd <path to POX>
root@poxlab:<path to POX># pyenv local pypy3.9-7.3.16

#Test if it works
root@poxlab:<path to POX># python3 --version
Python 3.9.19 (a2113ea87262, Apr 21 2024, 05:40:24)
[PyPy 7.3.16 with GCC 10.2.1 20210130 (Red Hat 10.2.1-11)]
```