1 Introduction

This is a repository to test a few algorithms to detect best sampling rate on a network for decision making. This is part of my graduate research thesis. As the testbench I will use OpenFlow, Mininet. References to software, tutorials, guides and paper I used will (not exhaustively) be listed below.

2 Description

Project is based in mininet + openVSwitch + Ryu Controller. Mininet is in charge of simulating the namespaces, openVSwitch receives rules for forwarding flows, Ryu Controller handles the communication between the logic layer and the forwarding layer. This is to say, the controller will observe the (possible multiple) switches in the network and will communicate with them new rules according to whatever criteria our logic sets.

2.1 More Specifically

(This might be incorrect as I change the topology and its surrounding environment quite frequently)

Topology is circular see (./py_utils/topologies/thick.py). When a hosts sends a packet to controller it will immediately bind its mac address to its ingress port. That way we can form a static map of where the mac addresses come from. With a map in place the controller can communicate with the switches the rules to set in terms of ethernet/mac addresses. Additionally, every rule will have appended to it a sub-rule to send statistics to collector at a specified rate.

(Comment: I am not sure yet whether to make the collector be the same as the controller)

3 TODOS

- ☒ Create Basic Topologies
- ☒ Setup Ryu Controller
  - ☒ Request Features
  - ☒ Learn Topology
  - ☒ Forward to Collector
- ☒ Setup Traffic
  - ☐ Setup simple Scapy Traffic (skip for now)
  - ☒ Setup Self-Similar Traffic
- ☐ Implement Time Sampling
- ☐ Create a `dpctl.c` modification to allow for sampling in the time domain.

4 References:

1. OpenFlow Software Switch(BOFUSS)
    1. Netbee
2. Mininet Simulator
3. Self-Similar Traffic