CPSDN Experiments

J. Rubio-Hernan M. El Barbori J. Garcia-Alfaro

February 10, 2018

1 Description

Programmable Networking to manage Cyber-Physical Systems communication.

Main Goal: Combine Programmable Networking technologies and control techniques from the physical domain to improve safety and security.

2 Use of Mininet

2.1 Summary

We propose an initial simulation testbed to verify the feasibility of our ongoing architecture. The simulation can scale up to to evaluate more complex simulation systems, e.g., to handle higher-order systems and verify the robustness of the approach.

2.2 Simulation Setup

- Installing the project:
  - Download Mininet Virtual Machine from `http://mininet.org/download/`.
  - Download topology.py and scada.zip files (cf. Sections 2.3 and 2.4 of this document).
  - Place file `Simple_Switch_13.py` into `/ryu/ryu/app` folder of Mininet's Virtual Machine.
- Running the project:
  - Run the Mininet topology as follows: `sudo ./topo.py`
  - Run SDN controllers as follows: `sudo ryu-manager simple_switch_13.py`
  - Run circuit areas (Host$_2$, Host$_3$, and Host$_4$) and mobile vehicles (Host$_5$, Host$_6$):
    * Host$_2$: `cd /scada/right_circuit/ and sudo ./rightcircuit.sh`
    * Host$_3$: `cd /scada/center_circuit/ and sudo ./centercircuit.sh`

    * Host$_4$: `cd /scada/left_circuit/ and sudo ./leftcircuit.sh`
    * Host$_5$: cd /scada/train1/ and sudo ./train1.sh
    * Host$_6$: cd /scada/train2/ and sudo ./train2.sh
  - Run the feedback controller (Host$_1$): `cd /scada/scada_controller/` and sudo `./scadacontroller.sh`
  - Order to input the IP address in the PS Controller (HMI)
    * IP: 10.0.0.2 (Right circuit)
    * IP: 10.0.0.3 (Center circuit)
    * IP: 10.0.0.4 (Left circuit)
    * IP: 10.0.0.5 (Train 1)
    * IP: 10.0.0.6 (Train 2)
    * IP: 10.0.0.1 (PS controller)

2.3 SCADA Protocols

The cyber-physical system represented in this simulation is a Railway circuit with two trains. The railway circuit has three different areas manages by three RTUs/PLCs (*Remote Terminal Units* and *Programmable Logic controller*) and all the system is managed by feedback Controller which holds the HMI (*Human Machine Interface*).

The circuit simulation and the mobile vehicles simulation are located in the following folders:

- Feedback controller: `/scada/scada_controller/scadacontroller.sh` (Runnig in Mininet ad Host$_1$)
- Right circuit: `/scada/right_circuit/rightcircuit.sh` (Running in Mininet as Host$_2$)
- Center circuit: `/scada/center_circuit/centercircuit.sh` (Running in Mininet Host$_3$)
- Left circuit: `/scada/left_circuit/leftcircuit.sh` (Running in Mininet as Host$_4$)
- Train 1: `/scada/train1/train1.sh` (Running in Mininet as Host$_5$)
- Train 2: `/scada/train2/train2.sh` (Runnimg in Mininet as Host$_6$)

2.4 TOPOLOGY

To create a network topology for our simulation using Mininet environment, we use a python code (`topology.py`) which calls the Mininet class from `mininet.net`.

- First, create an empty network using (`net = Mininet()`);
- Then, create the SDN controller for the network (`c0 = net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6633)`);

- We add Feedback controller to the network: `h1 = net.addHost('h1', ip="10.0.0.1/24")`
- We add right circuit to the network: `h2 = net.addHost('h2', ip="10.0.0.2/24")`
- We add center circuit to the network: `h3 = net.addHost('h3', ip="10.0.0.3/24")`
- We add left circuit to the network: `h4 = net.addHost('h4', ip="10.0.0.4/24")`
- We add train 1 to the network: `h5 = net.addHost('h5', ip="10.0.0.5/24")`
- We add train 2 to the network: `h6 = net.addHost('h6', ip="10.0.0.6/24")`
- We add switch 1 to the network: `s1 = net.addSwitch('s1')`
- We add switch 2 to the network: `s2 = net.addSwitch('s2')`
- We add switch 3 to the network: `s3 = net.addSwitch('s3')`
- We add switch 4 to the network: `s4 = net.addSwitch('s4')`
- We connect PS controller with switch 1: `net.addLink(h1, s1)`
- We connect switch 1 with switch 2: `net.addLink(s1, s2)`
- We connect switch 2 with switch 3: `net.addLink(s2, s3)`
- We connect switch 2 with switch 4: `net.addLink(s2, s4)`
- We connect right circuit with switch 3: `net.addLink(h2, s3)`
- We connect center circuit with switch 3: `net.addLink(h3, s3)`
- We connect left circuit with switch 3: `net.addLink(h4, s3)`
- We connect train 1 with switch 4: `net.addLink(h5, s4)`
- We connect train 2 with switch 4: `net.addLink(h6, s4)`
- We build the network: `net.build()`
- We run SDN controller with `c0.start()`
- We attach switch 1 to the SDN controller: `s1.start([c0])`
- We attach switch 2 to the SDN controller: `s2.start([c0])`
- We attach switch 3 to the SDN controller: `s3.start([c0])`
- We attach switch 4 to the SDN controller: `s4.start([c0])`

2.5 Additional Information

- To display the flow table in the switch, use the command: `ovs-ofctl dump-flows "switch name (e.g., s1)"`
- To clean the flow history, use command: `sudo mn -c cleanup`

2.6 Sample results

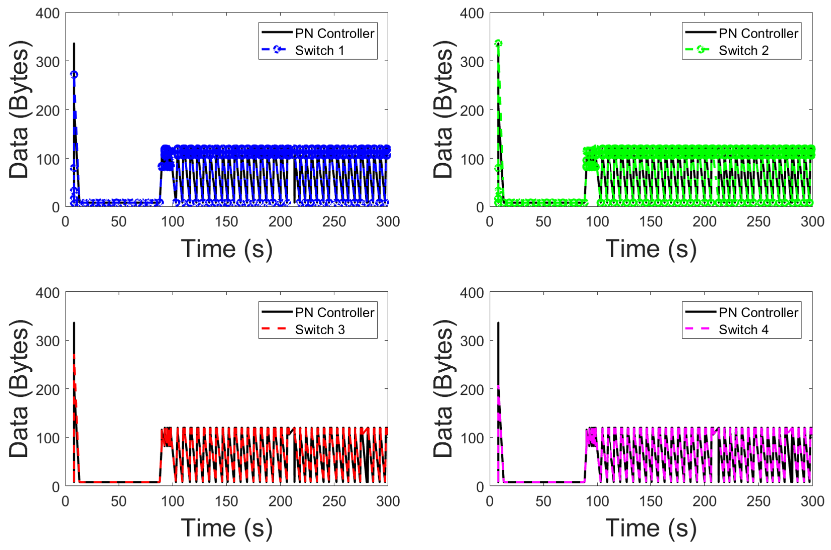

Figure 1: Probes sending all the data flows to the control domain.

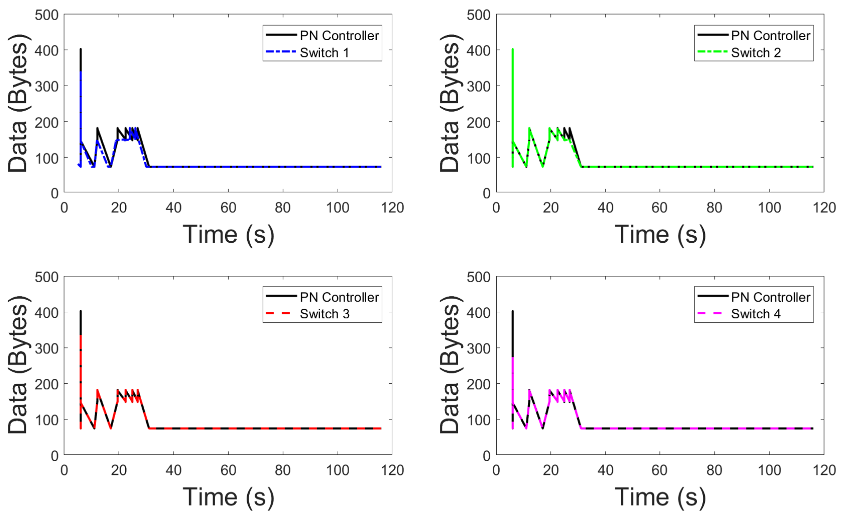

Figure 2: Effectors sending rules at the initialization of the network.