Computer Network Lab: Lab #3 (SDN w/ Mininet and Ryu)

塗大爲
Department of Computer Science and Information Engineering
National Taiwan University
b07902024@ntu.edu.tw
陳威翰
Department of Computer Science and Information Engineering
National Taiwan University
b07902132@ntu.edu.tw
黃于軒
Department of Computer Science and Information Engineering
National Taiwan University
b07902134@ntu.edu.tw
楊子平
Department of Computer Science and Information Engineering
National Taiwan University
b07902136@ntu.edu.tw

I. Environment

The Mininet suite and Ryu should run on most Linux distros with dependencies properly installed. For the sake of completeness, we briefly describe the environment in which we demonstrated our project.

- Operating System: Arch Linux (2021/05/25)
- Mininet: 2.3.0-2, installed from the Arch User Repositories[1]
- Ryu: 4.34, installed with `pip`

We provide two topologies: the tree topology and the data center topology. Both of them are placed in the `topo` directory and can be launched as follows.

- Tree: Use `n_hosts` to configure the number of hosts.

  ```
  sudo mn --custom topo/tree.py --topo=tree-
      topo,n_hosts=5 --controller=remote
  ```

- Data Center: Use `n_cores` to configure the number of core switches and `n_aggregations` for the number of aggregations.

  ```
  sudo mn --custom topo/data-center.py --
      topo=data-center-topo,n_cores=4,
      n_aggregations=4 --controller=remote
  ```

For the controller, whether to *drop* or *reroute* flows with excessive bandwidth can be configured by modifying the `congestion` key in `action.conf` (by setting it to either `drop` or `reroute`). The Ryu controller can then be invoked by

```
ryu-manager --observe-links controller/
    controller.py --config-file=action.conf
```

Note that the `--observe-links` option is crucial as we need to manually block/unblock ports based on the observed link structures.

II. Explain the pros and cons that there are loops in a network topology

An obvious advantage of having loops in the topology is the ability to re-route flows. If the topology is tree-like (acyclic), then the paths between hosts are uniquely determined, and there is not much to do when congestion happens. Having loops in the network means that there are more links to be leveraged, and more complex routing decisions can be made based on the bandwidths of the flows.

Broadcast storms (as described in the following section) is one major drawback of using a non-tree-like topology. A more sophisticated routing algorithm is also required in topologies involving loops to utilize all the links and balance the loads evenly.

III. Explain the broadcast storm and how you handle it in this lab

If the network topology contains cycles, broadcast storms may occur —that is, when a switch sends or forwards a broadcast frame, the frame may be routed along a cycle back to itself. The frame is thus sent and broadcast indefinitely.

[1] https://aur.archlinux.org/packages/mininet/

Our project handles this issue by monitoring the creation of links in the controller. We may then compute a spanning tree and only route traffic along the tree edges. Ports that correspond to non-tree edges are blocked using the `OFPPortMod` messages.

IV. Is there any better solution to handle broadcast storms under SDN? If yes, explain how; if no, explain why

A better and more standard way of handling this is STP (Spanning Tree Protocol). In STP, a root switch first is chosen. Then, the remaining switches communicate via Bridge Protocol Data Units (BPDUs) to compute the port with the lowest cost to the root switch. Unlike our method, the protocol can handle dynamic link changes and link failures. On the other hand, it is perhaps more difficult to perform per-flow rerouting with STP since the same spanning tree is used to transmit all data.