Software Defined Networking

Lab Work 2 Solution

Jeremias Blendin, Leonhard Nobach, Christian Koch,
Julius Rückert, Matthias Wichtlhuber

PS - Peer-to-Peer Systems Engineering Lab
Dept. of Electrical Engineering and Information Technology
Technische Universität Darmstadt
Rundeturmstr. 12, D-64283 Darmstadt, Germany
http://www.ps.tu-darmstadt.de/

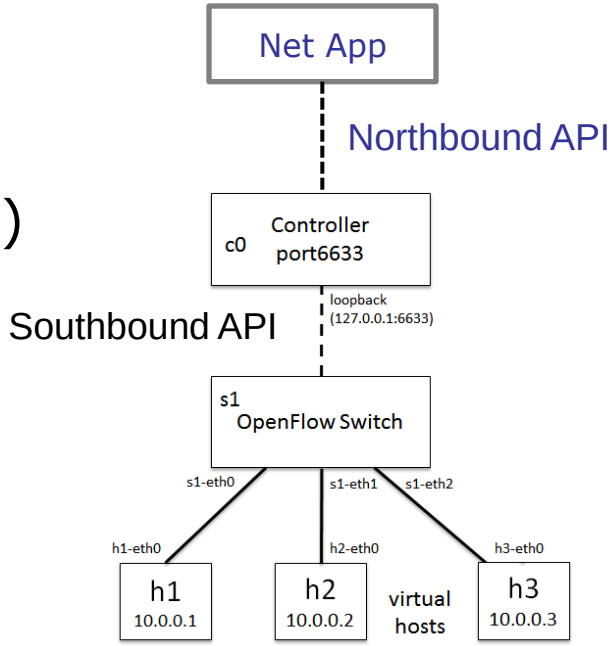
INTRODUCTION TO

OpenFlow Controller / RYU

Short Recap

TECHNISCHE UNIVERSITÄT DARMSTADT

- Previously we manually added rules in the switch
  - dpctl add-flow tcp:127.0.0.1:6634\
    in_port=1,idle_timeout=0,actions=output:2
- This should be done automatically
  - Task of a Network Application (NetApp)
  - E.g. a simple switching NetApp



[1] http://sdnhub.org/resources/useful-mininet-setups/

Installing RYU

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reboot your existing Mininet VM and enter:
  - sudo -s
  - apt-get install python-eventlet python-routes python-webob python-paramiko python-pip python-dev libxml2-dev libxslt-dev zlib1g-dev
  - pip install ryu
  - pip install six==1.8.0
  - mn -c

Run a simple_switch

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enter:
  - mn --topo single,3 --mac --arp --switch ovsk\
    --controller=remote,ip=127.0.0.1
  - h1 ping h2

- Open a second terminal and connect to the VM
- Copy the example app to your VM
  https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch.py
- Execute ryu-manager ./simple_switch.py
- Now the ping from terminal 1 succeeds

Understand how it works

TECHNISCHE UNIVERSITÄT DARMSTADT

- A step-by-step explanation can be found here
  - http://osrg.github.io/ryu-book/en/html/switching_hub.html
  - Read it carefully!
- Other resources like books and tutorials available
  - E.g. http://books.google.de/books?id=JC3rAgAAQBAJ

Task 1: Port Mirroring

TECHNISCHE UNIVERSITÄT DARMSTADT

- Modify the *simple_switch.py* in a way that all received ICMP request packets are sent through the two other *out_ports* of the switch. The packet should not be sent back to the port from where it originated.
  - The basis for the task is the Ryu application simple_switch.py and OF 1.0: https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch.py
  - **A ping request** from h1 to h2 should result in a ping reply to h1 from h2 and h3. As a result, h1 receives more packets then it has sent.
    - It is sufficient for the solution to work for in the example network with 3 hosts
    - Mininet provides a fixed mapping between OpenFlow port numbers, MAC, and IP addresses. This information should be used for implementation.
  - Carefully think about what actions need to be applied to the ICMP packets
  - Have a look at the respective standards documents:
    - OpenFlow Switch Specification 1.0.0 & Errata

    https://www.opennetworking.org/sdn-resources/technical-library

Solution: Terminal Output

TECHNISCHE UNIVERSITÄT DARMSTADT

Terminal1

```
mininet@mininet-vm:~$ sudo mn --topo single,3 --mac --arp --switch ovsk --
controller=remote,ip=127.0.0.1
```

Terminal2

```
mininet@mininet-vm:~$ sudo ryu-manager ./simple_switch_sol.py
loading app ./simple_switch_sol.py
loading app ryu.controller.ofp_handler
instantiating app ./simple_switch_sol.py of SimpleSwitchinstantiating app
ryu.controller.ofp_handler of OFPHandler
```

Terminal1

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.99 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.14 ms (DUP!)
```