Software Defined Networking

Lab Work 2 Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

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/

TECHNISCHE UNIVERSITÄT DARMSTADT

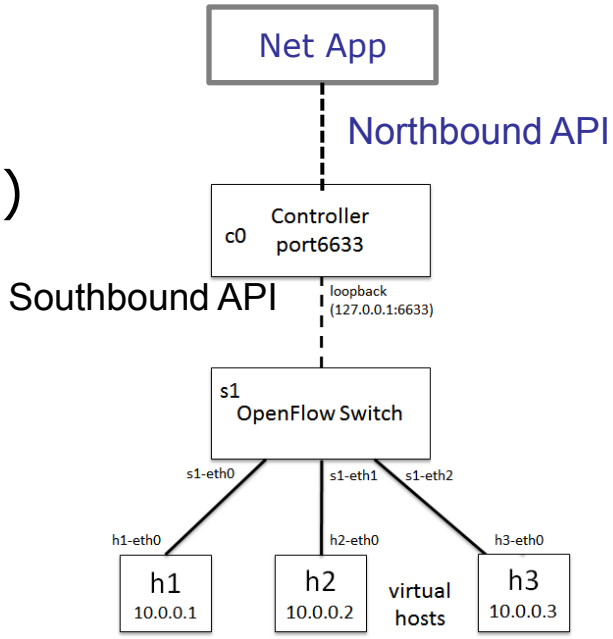
INTRODUCTION TO

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`
  - `$ 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 -> timeout`
- Open a second terminal and connect to the VM
  - Execute
    - `$ ryu-manager ryu.app.simple_switch --verbose`
  - h1 ping h2 ➡ success
  - Investigate the OpenFlow rules in switch s1
    - New tool: ovs-ofctl
    - `$ ovs-ofctl dump-flows s1`

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: Packet Replication 1/2

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

Task 1: Packet Replication 2/2

TECHNISCHE UNIVERSITÄT DARMSTADT

Debugging

- How to open a third terminal and connect it to one of the hosts?
  - Use xterm if you have a GUI installed
  - Open a second ssh session to the mininet VM
  - In Mininet run: `mininet> py h3.pid -> 3013`
  - Attach to h3 by running `$ sudo mnexec -a 3013 bash`
  - Verify by running `$ ip a ->` you should see the interfaces of h3 only
- Run tcpdump on h3
  - Open a terminal on h3
  - `$ tcpdump -eUvi h3-eth0`
  - `mininet> h1 ping h3`
  - With the packet duplication code in place
  - `mininet> h1 ping h2`
  - A packet copy of the ICMP request to h2 should be visible in the tcpdump output
    - Why does h3 not send an ICMP reply to the ICMP request?