

Software Defined Networking

Lab Work 3 Solution



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/>

Lab 3 Task 1

- ❖ Look at the source code of the Ryu modules `simple_switch.py` and `simple_switch_13.py`.
 - Describe and explain the differences between the OpenFlow 1.0 and OpenFlow 1.3 version of `simple_switch`.

Lab 3 Task 1 Example Solution

- ❖ Differences between `simple_switch.py` and `simple_switch_13.py`
 - Table miss
 - OpenFlow 1.0
 - Single table
 - Implicit rule to send packets to controller
 - OpenFlow 1.3
 - Multiple tables
 - Default behavior drops packet
 - For every new switch
 - Install rule with priority 0 that sends table misses to controller
 - Actions vs. Instructions
 - OpenFlow 1.0
 - Actions only
 - OpenFlow 1.3
 - Instructions and Actions
 - Use `OFPIT_APPLY_ACTIONS` to get the same behavior as in OpenFlow 1.0
 - Code improves handling of buffering for „packet_in“
 - OpenFlow 1.3 does differentiate between buffered packets and unbuffered packets

Lab 3 Task 2 Example Solution

❖ Create an OpenFlow 1.3 version called `simple_switch_filter_13.py`.

1. Get inspired by `simple_switch_13.py` and `simple_switch_filter.py`
2. Use two flow tables for the OpenFlow 1.3 version
 1. Use the first flow table for matching input ports and source MAC address

```
# Add rule for this mac at this port to go to table 1
match = parser.OFPMatch(in_port=in_port, eth_src=src)
instructions = [parser.OFPIInstructionGotoTable(1)]
self.add_flow(datapath, 200, match, instructions, None)

# Drop all other packets
instructions = []
match_other = parser.OFPMatch(in_port=in_port)
self.add_flow(datapath, 100, match_other, instructions, None)
```

2. Use the second flow table for sending the packets out the correct port.

```
actions = [parser.OFPACTIONOutput(out_port)]
instructions = [parser.OFPIInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
actions)]
self.add_flow(datapath, 1, match, instructions, None, 1)
```

Lab 3 Task 2 Example Solution

- ❖ Describe and discuss the differences regarding the number of flow rules used
 1. Consider different topologies and number of hosts for the sake of discussion even though running them with the `simple_switch_filter.py` is not possible
 2. Mathematically describe an upper bound for the number of flow rules
 1. For the OpenFlow 1.0 version: $n*(n-1)$ Flow Entries $\rightarrow O(n^2)$
 2. For the OpenFlow 1.3 version: $3*n$ Flow Entries $\rightarrow O(n)$
- ❖ Note that depending on your implementation the value of OpenFlow 1.3 might be slightly different
- ❖ Take-away message:
 - OpenFlow 1.0 with one flow table is not scalable for complex rulesets
 - OpenFlow > 1.1 is required for that, enables linear growth of number of rules in many cases