Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Oliver Gasser, M.Sc.
Cornelius Diekmann, M.Sc.

Master Course Computer Networks

Exercise 3
(submission until December 3th, 10:30 CET via SVN)
(submission of corrected version until December 6th, 10:30 CET via SVN)

Note1: Each subproblem gives you 0, 1 or 2 points. See the slides from October 29th for more information on the 0.3 bonus.
Note2: Subproblems marked by * can be solved without preceding results.

Understanding Software Defined Networking (The basics)

a)* What is the task of the forwarding plane?

b)* What is the task of the control plane?

c) Research task: What is the data plane?
(To get the points, you have to answer a, b, and c)

/ 2

d)* What is the definition of a software defined network? (2 aspects)

/ 2

e)* What are the three central abstractions in Software Defined Networking (according to our slides)?

f) What are the key benefits of these abstractions?

/ 2

g) You have 4 switches without forwarding logic, 4 common high-end CPUs, 4 ASICs, and enough spare parts to build arbitrary additional devices (if necessary). Distribute the 4 CPUs and ASICs to design a software defined network that obeys the basic SDN principle. Shortly outline the ASICs tasks.

/ 2

Controller programming (in our pseudo code)

You have an Ethernet switch with only Alice (IPv4 addr = 192.168.0.1) and Bob (IPv4 addr = 192.168.0.2) connected. Only Alice and Bob use the switch and they only communicate with each

other. Warning: they might use broadcasts! At the moment, Alice is at port 5, Bob at port 8. Below, you see an example code with a static flow table setup.
However, Alice and Bob often change positions (ports) and the code cannot handle this. They unplug their cable at the switch and plug it in later at some different port.
Write a controller that deals with this mobility.
You get the `link_down` function that is called whenever an Ethernet cable is unplugged from the switch. You can issue `send_flow_mod(sw, drop_all)` to drop all the switch's flow table entries. Set up flow table entries, when possible, to increase performance.
Hint1:
Verify that `ethernetType` is IPv4 (0x800) before checking for the IP address. If you try to check for an IPv4 address and the packet is in fact no IPv4 packet, you get undefined behavior! See this example:
```
{match_all with inPort = ?  and MACsrc = ?  and MACdst = ?  and ethernetType = 0x800 and
IPv4Dst = ?  and IPv4Src = ?}
```

Hint2:
Introduce some global variables.

Hint3:
Look at the slides. There, the match possibilities and actions for the flow table entries are defined. You can make use of any python-like pseudo code features, the only constraints are the OpenFlow constraints.

Hint4:
The example code does not work for the presented mobility scenario. You need to re-write it from scratch. Be aware that this is a very challenging task that requires some thought. We recommend that you practice by developing the simple and efficient learning switch from the slides first.

The non-mobility-aware code:

```
def link_down (sw: switchId, port: SwitchPort): unit =
  /* TODO */

def switch_connected (sw: switchId): unit =
  /* Alice to Bob */
  Match = {match_all with inPort = 5}
  send_flow_mod(sw, (add_flow 10 Match [Output Port(8)]))

  /* Bob to Alice */
  Match = {match_all with inPort = 8}
  send_flow_mod(sw, (add_flow 10 Match [Output Port(5)]))


def packet_in (sw: switchId, pk: packetIn): unit =
    /* I'm lazy, I rely on efficient flow tables. flood packets */
    send_packet_out(sw,{
      output_payload = pk.input_payload;
      apply_actions = [Output AllPorts]
    })
```

a)* Implement the controller correctly. Use the following template. Make sure the mobility requirement is fulfilled. You can reuse the lazy packet flooding. You can only get full points for performance if the controller operates correctly.

```
def link_down (sw: switchId, port: SwitchPort): unit =
```

```
def switch_connected (sw: switchId): unit =
```

```
def packet_in (sw: switchId, pk: packetIn): unit =
```

Your submission is graded as follows.

b) Correct implementation of mobility requirement, disregarding flow tables.

2

c) Overall correctness of solution, disregarding flow tables.

2

d) Efficient use of flow table entries

2

e) Overall correctness of complete solution.

2