

# Exercise for Lecture Software Defined Networking

Prof. Dr. David Hausheer, Julius Rückert

Christian Koch, Jeremias Blendin, Leonhard Nobach, Matthias Wichtlhuber



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Winter Term 2016/17

Exercise No. 7

Published: 17.01.2017

Submission exclusively via Moodle, Deadline: 24.01.2017

Contact: Please use the Moodle forum to post questions and remarks on the exercise.

Web: <http://www.ps.tu-darmstadt.de/teaching/ws1617/sdn/>

Submission: <https://moodle.tu-darmstadt.de/course/view.php?id=8385>

Surname (Nachname):	Chen
First name (Vorname):	Zhen
ID# (Matrikelnummer):	2665935

Team member: Letian Feng(2255840), Chunyuan Yu(2587628)

## Problem 7.1 - P4 Parse Graph

Given the following file headers.p4.

```
1
2 header_type ethernet_t {
3     fields {
4         bit<48> dstAddr;
5         bit<48> srcAddr;
6         bit<16> etherType;
7     }
8 }
9
10 header_type vlan_t {
11     fields {
12         bit<16> vlanTag;
13         bit<16> etherType;
14     }
15 }
16
17 parser start {
18     return parse_ethernet;
19 }
20
21 header ethernet_t ethernet;
22
23 #define ETHERTYPE_VLAN 0x8100
24
```

---

```
25 parser parse_ethernet {
26     extract(ethernet);
27     return select(latest.etherType) {
28         ETHERTYPE_VLAN : parse_vlan;
29         default: ingress;
30     }
31 }
32
33 header vlan_t vlan;
34
35 parser parse_vlan {
36     extract(vlan);
37     return ingress;
38 }
```

---

**a) Explain the general purpose of the P4 code above, and describe the result of the code applied to an incoming packet.**

---

The general purpose of the code is to check if the incoming packet is a vlan-tagged packet, if yes then get the vlan-tag of the packet and do some work defined in control "ingress". For example, a packet vlan-tagged with vlan-5 will be first parsed in "parse\_ethernet" and be selected as "ETHERTYPE\_VLAN", then the parser "parse\_vlan" will be executed and all the information of header\_type "vlan\_t" will be extracted and then saved in the header "vlan" waiting for the processing in the "ingress".

---

**b) What does the statement `return ingress;` do?**

---

Ingress will check if the header "vlan" is valid, if yes then do some other processing such as untag the packet and forward it to a port, or change the tag etc. Otherwise, the ingress could add a vlan-tag on the packet or do some other works.

---

## Problem 7.2 - P4 Matchers and Actions

---

Given the following P4 code.

```
1 #include "headers.p4" //this is the file in the last problem
2                               // import program in problem 7.1
3 header_type ingress_md_t {    // define header type ingress_md_t, which has a field vlanid
4     fields {
5         bit<16> vlanid;
6     }
7 }
8
9 metadata ingress_md_t ingress_md; // declare a metadata named ingress_md in type of ingress_md_t
10
11 action _nop() { }             // do nothing
12
13 // -----
14
15 table map_vlan { // table for vlanid, read ingress port & vlanid in full length and save in the table;
16     reads {       // table can do actions: sendPushTag, sendChangeTag, sendPopTag, send or _drop
17         standard_metadata.ingress_port : exact;
18         ingress_md.vlanid : exact;
19     }
20     actions {
21         sendPushTag;
22         sendChangeTag;
23         sendPopTag;
24     }
25     send;
26     _drop;
27 }
28
29 action sendPushTag(in bit<16> tag, in bit<16> port) { // add parameter "tag" to the
30     add_header(vlan);                                // vlan header and send it out
31     // Missing code here                             // via parameter "port"
32 }
33
34 action sendChangeTag(in bit<16> tag, in bit<16> port) { // change the tag of vlan
35     vlan.vlanTag = tag;                                // header to the parameter
36     standard_metadata.egress_spec = port;              // "tag" and send it out via
37 }                                                       // parameter "port"
38
39 action sendPopTag(in bit<16> port) { // set the packet non-vlan-tagged & remove the vlan-tag
40     ethernet.etherType = vlan.etherType; // and send it out via parameter "port"
41     remove_header(vlan);
42     standard_metadata.egress_spec = port;
43 }
44
45 action send(in bit<16> port) { // set egress data in the metadata as the parameter "port"
46     standard_metadata.egress_spec = port;
47 }
```

---

```
48
49 action _drop() {                                     // drop the packet
50     drop();
51 }
52
53 // -----
54
55 action set_has_vlan() {                               // set the vlanid in the metadata as in vlan header
56     ingress_md.vlanid = vlan.vlanTag;
57 }
58
59 action set_has_no_vlan() {                           // set vlanid in the the metadata as 0
60     ingress_md.vlanid = 0;
61 }
62
63
64 table prepare {    // table for preparation, read the etherType in full length and save in the table,
65     reads {        // the table can do actions: set_has_vlan, set_has_no_vlan
66         ethernet.etherType : exact;
67     }
68     actions {
69         set_has_vlan;
70         set_has_no_vlan;
71     }
72 }
73
74 control ingress { // call the table "prepare" & "map_vlan" to read the packet and do relevant works
75     apply(prepare);
76     apply(map_vlan);
77 }
78
79 control egress { }                                     // do nothing
```

---

a) Describe the semantics of the code above. Be precise.

---

As shown above, in green.

---

## b) Complete the missing code in the sendPushTag action.

---

```
action sendPushTag ( in bit <16> tag , in bit <16> port ) {  
    add_header(vlan);  
    ethernet.etherType = ETHERTYPE_VLAN;  
    standard_metadata.egress_spec = port ;  
}
```

---

## c) P4 Pipelines and Flow Entries

---

Assume the P4 code is installed on a switch with 4 ports (Port 0-3). Port 0 is designated as a tagged trunk port, Port 1,2,3 are designated as untagged access ports. Packets coming in on any of the ports 1,2 or 3 shall be forwarded on Port 0, tagged with the respective VLAN 1,2, or 3. Likewise, packets coming in on Port 0 and being tagged with the VLAN 1,2 or 3 must be forwarded untagged on the respective port 1,2, or 3.

**Create a set of flow entries for the tables prepare and map\_vlan that ensure the switch behaves as desired.**

Use an informal notation like "match=x:1,y:2, action=foobar"

```
match=ethernet.etherType:0x8100, standard_metadata.ingress_port:0, ingress_md.vlanid:1,  
action=sendChangeTag(0,1);  
match=ethernet.etherType:0x8100, standard_metadata.ingress_port:0, ingress_md.vlanid:2,  
action=sendChangeTag(0,2);  
match=ethernet.etherType:0x8100, standard_metadata.ingress_port:0, ingress_md.vlanid:3,  
action=sendChangeTag(0,3);  
match=ethernet.etherType:0x0800, standard_metadata.ingress_port:1, action=sendPushTag(1,0);  
match=ethernet.etherType:0x0800, standard_metadata.ingress_port:2, action=sendPushTag(2,0);  
match=ethernet.etherType:0x0800, standard_metadata.ingress_port:3, action=sendPushTag(3,0);
```