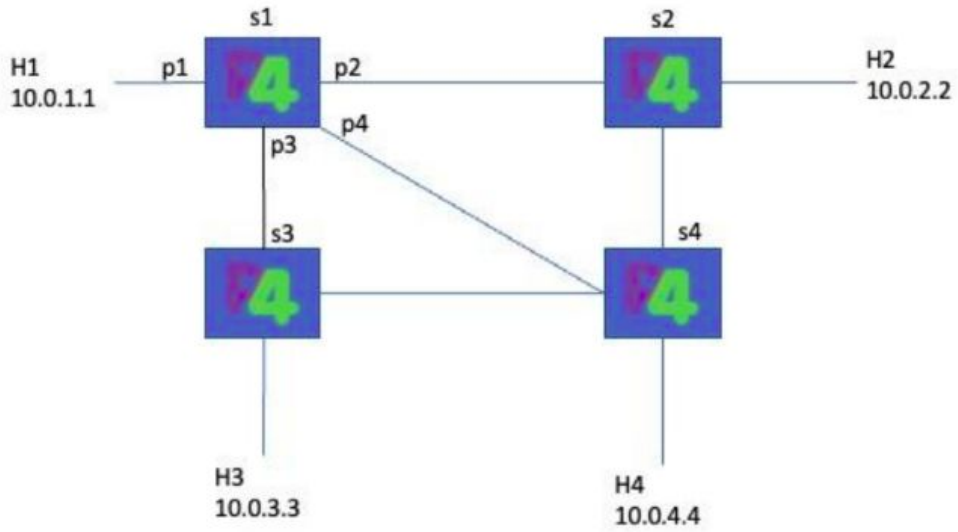
Load Balancer
Using P4 and BMV2 hash extern

(https://github.com/Tigohsr/loadBalancer-P4)

Felipe Novais
Thiago Henrique S Rodrigues

topo.txt

```
switches 4
hosts 4
h1 s1
s1 s2
s1 s3
s1 s4
s2 h2
s2 s4
s3 h3
s3 s4
s4 h4
~
~
topo.txt
```



topo.py

```
105     sw_addr = ["10.0.%d.254" % n for n in xrange(nb_hosts)]
```

```
59          for h in xrange(nb_hosts):
60              host = self.addHost('h%d' % (h + 1),
61                                  ip = "10.0.%d.%d/24" % (h+1, h+1),
62                                  mac = '00:04:00:00:00:%02x' % h)
63
```

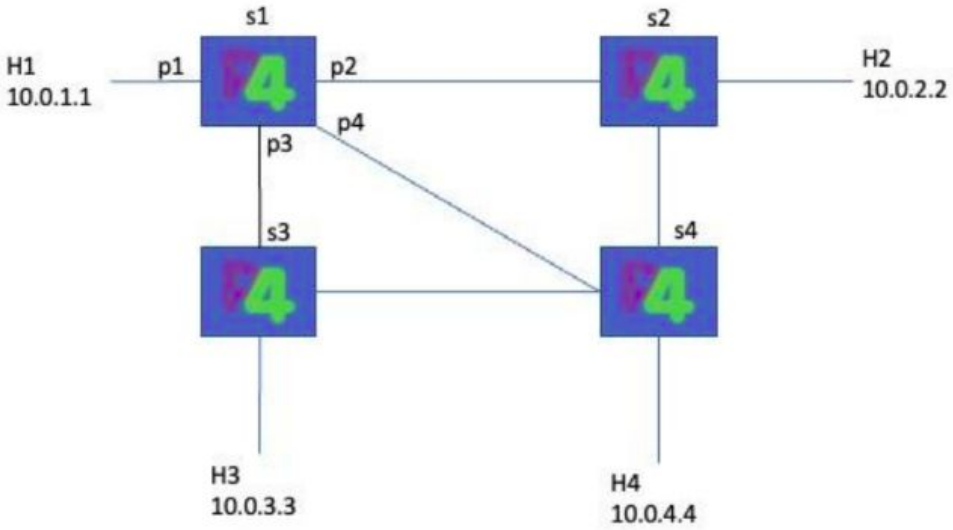
```
178         if (i == 0):
179                 print "Running lb_s1.txt"
180                 cmd = [args.cli, "--json", args.json,
181                        "--thrift-port", str(_THRIFT_BASE_PORT + i)]
182                 with open("lb_s1.txt", "r") as f:
183                     print " ".join(cmd)
184                     try:
185                         output = subprocess.check_output(cmd, stdin = f)
186                         #print ("Debugging switch match-action table")
187                         print output
188                     except subprocess.CalledProcessError as e:
189                         print e
190                         print e.output
191                 print "**********"
```

Tables in Control Plane

```
table_set_default ipv4_lpm drop
table_set_default lb_hash_exact drop
~
~
~
command_s1.txt                                                1,1           All
table_set_default ipv4_lpm drop
table_add ipv4_lpm ipv4_forward 10.0.2.2/32 => 00:04:00:00:00:01 2
~
~
command_s2.txt                                                1,1           All
table_set_default ipv4_lpm drop
table_add ipv4_lpm ipv4_forward 10.0.3.3/32 => 00:04:00:00:00:02 2
~
~
command_s3.txt                                                1,1           All
table_set_default ipv4_lpm drop
table_add ipv4_lpm ipv4_forward 10.0.4.4/32 => 00:04:00:00:00:03 4
~
~
command_s4.txt                                                1,1           All
```

ports.py 1/2

```python
import os, binascii, sys

# File that will store control plane table entries
s1_rules = open('lb_s1.txt', 'w')

# IPs in hex
h1 = '0a000101' # 10.0.1.1
s1 = '0a0001fe' # 10.0.1.254

h2 = '0a000202' # 10.0.2.2
h3 = '0a000303' # 10.0.3.3
h4 = '0a000404' # 10.0.4.4

# Original locations for destination from h1 to s1
orig_destinations = [h2, h3, h4]

# Original egress ports from h1 to s1
s1_ports = [2, 3, 4]

# TCP ports from 1000 to 1499
dports = range(1000,1500)

# Fixed source port to 1234
sport = '04d2' # 1234

tcp_proto = '06' # IP Protocol

ports = []
rules = []
```



ports.py 2/2

```
31 print(" **** Generating rules for load balancer on S1... **** ")
32 for dport in dports:
33     # Initialize copy of original destination and ports on s1
34     if len(ports) == 0:
35         ports = s1_ports.copy()
36         destinations = orig_destinations.copy()
37
38     # Cryptographically secure random function to select a random port on s1
39     port = int(binascii.hexlify(os.urandom(2 ** 16)), 16) % len(ports)
40
41     # Pop port from list to provide a certain fairness during port selection
42     s1_port = str(ports.pop(port))
43     destination = destinations.pop(port)
44
45     # TCP-tuple that will be hashed
46     tcp_tuple = h1 + s1 + sport + format(dport,'0>4x') + tcp_proto
47
48     # CRC32 hashing the TCP-tuple
49     crc32_hash = str(binascii.crc32(binascii.a2b_hex(tcp_tuple)))
50
51     # Append to a list the rule that will be an entry on the control plane
52     rules.append('table_add lb_hash_exact lb_hash_forward ' + crc32_hash + '
    => 0x' + destination + ' ' + s1_port)
53
54
55 # Writing list to file
56 s1_rules.write('\n'.join(rule for rule in rules))
57
58 print(" **** Generated rules for load balancer on S1 **** ")
```

```
Iteration #1:
[  2,  3,  4]
[ h2, h3, h4]

Port - randomized index:  2

Iteration #2:

[  2,  4]
[ h2, h4]

Port - randomized index:  1

Iteration #3:

[ 4]
[h4]

Port - randomized index: 1

Iteration #4:
[  2,  3,  4]
[ h2, h3, h4]

…
```

run_task.sh

```
THIS_DIR=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )

source $THIS_DIR/../env.sh

P4C_BM_SCRIPT=/usr/local/bin/p4c/build/p4c

SWITCH_PATH=/usr/local/bin/simple_switch

CLI_PATH=$BMV2_PATH/tools/runtime_CLI.py
#
$P4C_BM_SCRIPT --target bmv2-ss-p4org -x p4-16 p4src/assignment.p4 -o p4prog
# This gives libtool the opportunity to "warm-up"
 sudo $SWITCH_PATH >/dev/null 2>&1
 sudo PYTHONPATH=$PYTHONPATH:$BMV2_PATH/mininet/ python3 ports.py
 sudo PYTHONPATH=$PYTHONPATH:$BMV2_PATH/mininet/ python topo.py \
     --behavioral-exe $SWITCH_PATH \
     --json p4prog/assignment.json \
     --cli $CLI_PATH
~
~
```

Generated Control Plane (lb_s1.txt)

```
table_add       lb_hash_exact   lb_hash_forward 451078402 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 835697345 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 685017984 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 1737825607 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 2123246598 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 1436778437 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 1287147140 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 2179088410 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 2566484315 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 3017024152 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 2865697753 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 3851322654 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 4237669471 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 3619219356 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 3466843869 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 1228655122 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 1344338771 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 2064492688 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 1645648337 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 760680214 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 877411927 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 526478740 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 108682453 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 2417080098 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 2299168355 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 2720519584 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 3141330145 => 0x0a000303 3
table_add       lb_hash_exact   lb_hash_forward 4101861926 => 0x0a000202 2
table_add       lb_hash_exact   lb_hash_forward 3982902119 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 3326808228 => 0x0a000404 4
table_add       lb_hash_exact   lb_hash_forward 3746570725 => 0x0a000303 3
```

Defines and types

```
 6 const bit<16> TYPE_IPV4 = 0x800;
 7
 8 // Max integer for a 32bits variable
 9 #define MAX_32BITS 4294967295
10
11 // IP protocol field that represents TCP
12 #define TCP_PROTO 6
13
14 // Device IP in hex
15 #define S1 0x0a0001fe
```

```
49 header tcp_t {
50     bit<16> srcPort;
51     bit<16> dstPort;
52     bit<32> seqNo;
53     bit<32> ackNo;
54     bit<4>  dataOffset;
55     bit<3>  res;
56     bit<3>  ecn;
57     bit<6>  ctrl;
58     bit<16> window;
59     bit<16> checksum;
60     bit<16> urgentPtr;
61 }
62
63 struct metadata {
64     bit<32>    hashedTuple;
65 }
```

Parsers

```
87      state parse_ethernet {
88          packet.extract(hdr.ethernet);
89          transition select(hdr.ethernet.etherType) {
90              TYPE_IPV4: parse_ipv4;
91              default: accept;
92          }
93      }
94
95
96      state parse_ipv4 {
97          packet.extract(hdr.ipv4);
98          transition select(hdr.ipv4.protocol) {
99              TCP_PROTO: parse_tcp;
100             default: accept;
101         }
102     }
103
104     state parse_tcp {
105         packet.extract(hdr.tcp);
106         transition accept;
107     }
```

Tables in P4

```
145     table ipv4_lpm {
146         key = {
147             hdr.ipv4.dstAddr: lpm;
148         }
149         actions = {
150             ipv4_forward;
151             drop;
152             NoAction;
153         }
154         size = 1024;
155         default_action = NoAction();
156     }
157
158     table lb_hash_exact {
159         key = {
160             meta.hashedTuple: exact;
161         }
162         actions = {
163             lb_hash_forward;
164             drop;
165             NoAction;
166         }
167         size = 1024;
168         default_action = NoAction();
169     }
```

Apply

```
171     apply {
172         if (hdr.ipv4.isValid()) {
173             // Checks if dstAddr is S1
174             if (hdr.ipv4.dstAddr == S1) {
175                 // Hashing TCP-tuple using CRC32
176                 hash(meta.hashedTuple,
177                      HashAlgorithm.crc32,
178                      (bit<32>)0,
179                      {hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, hdr.tcp.srcPort, hdr.tcp.dstPort, hdr.ipv4.protocol},
180                      (bit<32>)MAX_32BITS);
181                 lb_hash_exact.apply();
182             } else {
183                 ipv4_lpm.apply();
184             }
185         } else {
186             drop();
187         }
188     }
189 }
```

output
hash algorithm
minimum value
data
maximum value

```
// Hashing TCP-tuple using CRC32
hash(meta.hashedTuple,
     HashAlgorithm.crc32,
     (bit<32>)0,
     {hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, hdr.tcp.srcPort, hdr.tcp.dstPort, hdr.ipv4.protocol},
     (bit<32>)MAX_32BITS);
```

Atividade 2 - Balanceador de Carga.pdf:

- um fluxo é caracterizado pelos 5 campos: IP origem, IP destino, porta origem, porta destino e protocolo;

Actions

```
132     action lb_hash_forward(ip4Addr_t dstAddr, egressSpec_t port) {
133         standard_metadata.egress_spec = port;
134         hdr.ipv4.dstAddr = dstAddr;
135         hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
136     }
137
138     action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
139         standard_metadata.egress_spec = port;
140         hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
141         hdr.ethernet.dstAddr = dstAddr;
142         hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
143     }
```

send.py

```
27 def main():
28
29     if len(sys.argv)<1:
30         print 'pass 1 arguments: <destination_port>'
31         exit(1)
32
33     s1 = "10.0.1.254"
34     addr = socket.gethostbyname(s1)
35     iface = get_if()
36     print "sending on interface %s to %s on dport: %s" % (iface, str(addr), sys.argv[1])
37
38     pkt =  Ether(src=get_if_hwaddr(iface), dst='ff:ff:ff:ff:ff:ff');
39
40     pkt = pkt / IP(src=get_if_addr(iface), dst=addr) / TCP(dport=int(sys.argv[1]), sport=1234)
41
42     pkt.show2()
43     sendp(pkt, iface=iface, verbose=False)
44
45
46 if __name__ == '__main__':
47     main()
```

DEMO

Thank you!

Source code: https://github.com/Tigohsr/loadBalancer-P4