SDN-NFV

lab4-report

111550120 吳盈庭

1. when s1-s2 link is up

a. check if traffic from h1 to h2 goes through s2 and s3

I ran the second command of the screenshot after sending packets from h1 (udp client) to h2 (udp server).

Compare the results of the two command:

- port 1: rx increases, since h1 is connected to s1 on port 1
- port 2: tx increases. When the link between s1 s2 is up, group bucket 1 is triggered and its treatment is to send the packet out to port 2.
- port 3: rx increases because h2's reply goes through s1's port 3.

```
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s1
OFPST_PORT reply (OF1.4) (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=2158.655s
  port  "s1-eth1": rx pkts=29, bytes=1909, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1387, bytes=191664, drop=0, errs=0, coll=0
           duration=2159.664s
  port  "s1-eth2": rx pkts=1384, bytes=191470, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1394, bytes=191933, drop=0, errs=0, coll=0
           duration=2159.677s
  port  "s1-eth3": rx pkts=1385, bytes=191540, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1384, bytes=191470, drop=0, errs=0, coll=0
           duration=2159.654s
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s1
OFPST_PORT reply (OF1.4) (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=2214.712s
  port  "s1-eth1": rx pkts=31, bytes=2026, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1423, bytes=196668, drop=0, errs=0, coll=0
           duration=2215.722s
  port  "s1-eth2": rx pkts=1423, bytes=196822, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1432, bytes=197054, drop=0, errs=0, coll=0
           duration=2215.735s
  port  "s1-eth3": rx pkts=1421, bytes=196544, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=1420, bytes=196474, drop=0, errs=0, coll=0
           duration=2215.711s
```

i. When the link between s1 and s2 is up, the first bucket in the group will be triggered since its watch port is on the link as shown in the screenshot.

```
ytw@root > groups                                                    19:12:01
deviceId=of:0000000000000001, groupCount=1
   id=0x1, state=ADDED, type=FAILOVER, bytes=94, packets=2, appId=nycu.winla
b.groupmeter, referenceCount=1
       id=0x1, bucket=1, bytes=94, packets=2, weight=-1, actions=[OUTPUT:2]
       id=0x1, bucket=2, bytes=0, packets=0, weight=-1, actions=[OUTPUT:3]
deviceId=of:0000000000000002, groupCount=0
deviceId=of:0000000000000003, groupCount=0
deviceId=of:0000000000000004, groupCount=0
deviceId=of:0000000000000005, groupCount=0
ytw@root >                                                           19:14:56
```

ii. Since bucket 1 is triggered, the packet goes out from port 2 and is sent to s2, and there should be an intent from s2 to h2.

```
19:13:56.121 INFO [AppComponent] targetIp = 10.6.1.2
19:13:56.155 INFO [AppComponent] recieved arp request for ip `10.6.1.2`
19:13:56.174 INFO [AppComponent] sent arp reply for h2's mac
19:13:56.221 INFO [AppComponent] intent!!!
19:13:56.361 INFO [AppComponent] Intent `of:0000000000000002`, port `1` => `
of:0000000000000005`, port `1` is submitted
```

Intents (1 total)

APPLICATION ID	KEY	TYPE	PRIORITY	STATE
176 : nycu.winlab.groupmeter	0x0	PointToPointIntent	100	Installed

iii. The intent is submitted, so there should be flow rules installed on the switches on the path. Take s3 as an example, the flow with app nme *net.intent is the one set by intent service.

Flows for Device of:0000000000000003 (6 Total)

Search | All Fields

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	1,173	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,172	5	0	ETH_TYPE:arp, ARP_OP:1	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,191	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1	1,015	100	0	IN_PORT:1, ETH_DST:00:00:00:00:00:02	imm[OUTPUT:2], cleared:false	*net.intent
Added	765	1,191	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	765	1,191	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

b. check if traffic from h2 to h1 goes through s?

Compare the result of the two commands:

- port 1: tx increases, because h2's reply is sent to h1 from s4's port 1.
- port 2: rx increases, because h2's reply is sent to s4's port 2 by s5.

```
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s4
OFPST_PORT reply (OF1.4) (xid=0x2): 3 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=4104.950s
  port  "s4-eth1": rx pkts=2638, bytes=365675, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=2640, bytes=365791, drop=0, errs=0, coll=0
           duration=4105.080s
  port  "s4-eth2": rx pkts=2641, bytes=365929, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=2638, bytes=365675, drop=0, errs=0, coll=0
           duration=4105.052s
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s4
OFPST_PORT reply (OF1.4) (xid=0x2): 3 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=4118.333s
  port  "s4-eth1": rx pkts=2648, bytes=367065, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=2660, bytes=367624, drop=0, errs=0, coll=0
           duration=4118.464s
  port  "s4-eth2": rx pkts=2659, bytes=367484, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=2648, bytes=367065, drop=0, errs=0, coll=0
           duration=4118.436s
```

h2 sends packets to h1 with point to point intent, and goes through s4. This can also be verified by the log message and flow on s4 as shown in the screenshots below.

```
20:18:27.615 INFO [AppComponent] intent!!!
20:18:27.635 INFO [AppComponent] Intent `of:0000000000000005`, port `1` => `
of:0000000000000001`, port `1` is submitted
```

Flows for Device of:0000000000000004 (7 Total)

Search | All Fields

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	4,169	40000	0	ETH_DST:00:00:00:00:00:02	imm[OUTPUT:2], metered:METER:1, cleared:false	nycu.winlab.groupmeter
Added	0	4,207	5	0	ETH_TYPE:arp, ARP_OP:1	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	4,226	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	4,208	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	11	179	100	0	IN_PORT:2, ETH_DST:00:00:00:00:00:01	imm[OUTPUT:1], cleared:false	*net.intent
Added	2,706	4,226	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,706	4,226	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

2. when s1-s2 link is down

   a. check if both traffic goes through s4

      When the link between s1 and s2 is down, since the watch port for bucket 1 is disabled, bucket 2 should be triggered, sending the packet to port 3 on s1. As shown in the screenshot, there are significantly more packets on s1's port 3, indicating the packets are sent out from port 3 to s4 when group bucket 2 is triggered.

```
mininet> link s1 s2 down
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s1
OFPST_PORT reply (OF1.4) (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=273.060s
  port  "s1-eth1": rx pkts=18, bytes=1376, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=192, bytes=25821, drop=0, errs=0, coll=0
           duration=273.138s
  port  "s1-eth2": rx pkts=21, bytes=2651, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=21, bytes=2651, drop=0, errs=0, coll=0
           duration=273.148s
  port  "s1-eth3": rx pkts=188, bytes=25529, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=190, bytes=25645, drop=0, errs=0, coll=0
           duration=273.129s
mininet>  sh ovs-ofctl dump-ports -O OpenFlow14 s4
OFPST_PORT reply (OF1.4) (xid=0x2): 3 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
           duration=399.112s
  port  "s4-eth1": rx pkts=272, bytes=37043, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=271, bytes=36997, drop=0, errs=0, coll=0
           duration=399.195s
  port  "s4-eth2": rx pkts=271, bytes=36997, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=270, bytes=36765, drop=0, errs=0, coll=0
           duration=399.188s
```

   b. check if iperf traffic rate is limited

      By the command " sh ovs-ofctrl dump-flows -O Openflow14 s4, we can make sure that the meter is successfully built on s4.

```
mininet>  sh ovs-ofctl dump-flows -O OpenFlow14 s4
 cookie=0x10000810531f2, duration=1085.819s, table=0, n_packets=0, n_bytes=0, send_flow_
rem priority=5,ip actions=CONTROLLER:65535,clear_actions
 cookie=0x10000261e0911, duration=1085.816s, table=0, n_packets=691, n_bytes=96049, send
_flow_rem priority=40000,dl_type=0x88cc actions=CONTROLLER:65535,clear_actions
 cookie=0x10000687421fe, duration=1085.727s, table=0, n_packets=691, n_bytes=96049, send
_flow_rem priority=40000,dl_type=0x8942 actions=CONTROLLER:65535,clear_actions
 cookie=0x1000011326ce6, duration=1085.726s, table=0, n_packets=0, n_bytes=0, send_flow_
rem priority=40000,arp actions=CONTROLLER:65535,clear_actions
 cookie=0x400007bf99f46, duration=1081.152s, table=0, n_packets=26, n_bytes=1228, send_f
low_rem priority=40000,dl_dst=00:00:00:00:00:02 actions=meter:1,output:"s4-eth2"
 cookie=0xae0000a497817c, duration=1083.245s, table=0, n_packets=0, n_bytes=0, send_flow
_rem priority=100,in_port="s4-eth1",dl_dst=00:00:00:00:00:02 actions=output:"s4-eth2"
 cookie=0x10000886ab9d5, duration=1085.817s, table=0, n_packets=0, n_bytes=0, send_flow_
rem priority=5,arp,arp_op=1 actions=CONTROLLER:65535,clear_actions
```