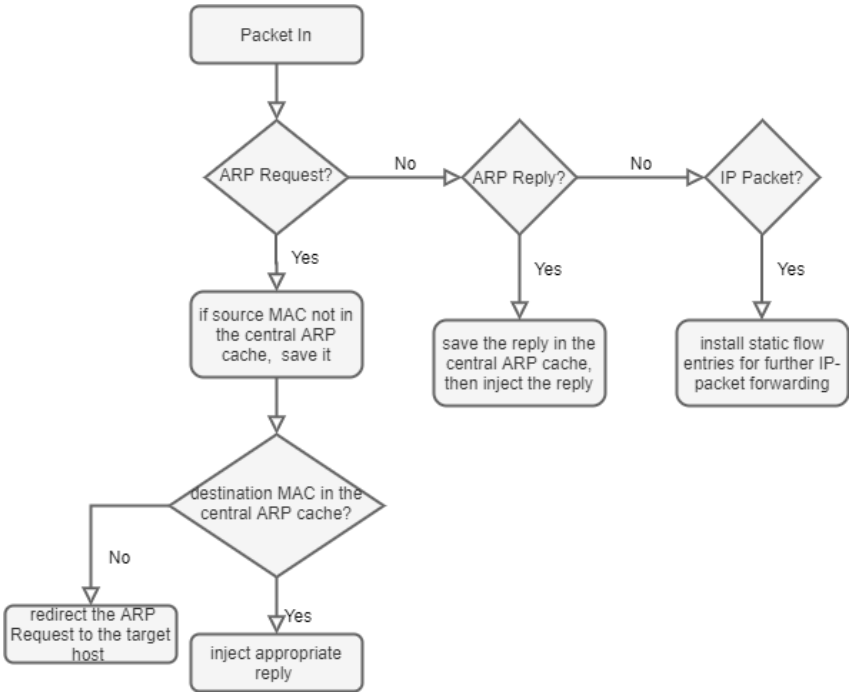
Software Defined Networking
Assignment 03

Peipei He, st169741@stud.uni-stuttgart.de, 3442500
Huicheng Qian, st169665@stud.uni-stuttgart.de, 3443114
Kuang-Yu Li, st169971@stud.uni-stuttgart.de, 3440829

Task1

The flow chart below illustrates how ARP requests/replies and IP packets are processed by the module ARPHandler.



Start Mininet with our custom topology (task14topo.py) without the `--mac` option.

```
student@sdnfp04:~$ sudo mn --switch ovsk --controller remote,port=6653 --custom ~/ex3/task14topo.py --topo task14topo
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
*** Adding hosts:
h11 h12 h13 h21 h22 h23 h41 h42 h43
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h11, s1) (h12, s1) (h13, s1) (h21, s2) (h22, s2) (h23, s2) (h41, s4) (h42, s4) (h43, s4) (s1, s2) (s2, s3) (s3, s4)
*** Configuring hosts
h11 h12 h13 h21 h22 h23 h41 h42 h43
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

ARP caches of all hosts were empty initially and MAC addresses were randomly assigned to these hosts.

```
mininet> h11 arp -n
mininet> h12 arp -n
mininet> h13 arp -n
mininet> h21 arp -n
mininet> h22 arp -n
mininet> h23 arp -n
mininet> h41 arp -n
mininet> h42 arp -n
mininet> h43 arp -n
```

```
mininet> py h11.MAC()
62:44:46:4f:e6:8f
mininet> py h12.MAC()
12:6d:5a:c6:1d:90
mininet> py h13.MAC()
8e:7c:ba:25:17:32
mininet> py h21.MAC()
c2:69:7b:96:0d:37
mininet> py h22.MAC()
86:5f:bb:aa:c2:b8
mininet> py h23.MAC()
fa:d8:ac:cc:84:99
mininet> py h41.MAC()
c2:9e:11:81:bf:0f
mininet> py h42.MAC()
d2:5e:cb:b1:02:99
mininet> py h43.MAC()
72:60:04:c1:60:e9
```

Start the ARPHandler module by running *Task31.launch* and run `Pingall` to test the connectivity between all hosts.

```
mininet> pingall
*** Ping: testing ping reachability
h11 -> h12 h13 h21 h22 h23 h41 h42 h43
h12 -> h11 h13 h21 h22 h23 h41 h42 h43
h13 -> h11 h12 h21 h22 h23 h41 h42 h43
h21 -> h11 h12 h13 h22 h23 h41 h42 h43
h22 -> h11 h12 h13 h21 h23 h41 h42 h43
h23 -> h11 h12 h13 h21 h22 h41 h42 h43
h41 -> h11 h12 h13 h21 h22 h23 h42 h43
h42 -> h11 h12 h13 h21 h22 h23 h41 h43
h43 -> h11 h12 h13 h21 h22 h23 h41 h42
*** Results: 0% dropped (72/72 received)
```

Now ARP caches of all hosts have been filled.

```
mininet> h11 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h11-eth0
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h11-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h11-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h11-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h11-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h11-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h11-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h11-eth0
mininet> h12 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h12-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h12-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h12-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h12-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h12-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h12-eth0
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h12-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h12-eth0
mininet> h13 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h13-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h13-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h13-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h13-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h13-eth0
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h13-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h13-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h13-eth0
```

```
mininet> h21 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h21-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h21-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h21-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h21-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h21-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h21-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h21-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h21-eth0
mininet> h22 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h22-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h22-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h22-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h22-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h22-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h22-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h22-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h22-eth0
mininet> h23 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h23-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h23-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h23-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h23-eth0
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h23-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h23-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h23-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h23-eth0
```

```
mininet> h41 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h41-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h41-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h41-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h41-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h41-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h41-eth0
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h41-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h41-eth0
mininet> h42 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h42-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h42-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h42-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h42-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h42-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h42-eth0
10.10.4.3                ether   72:60:04:c1:60:e9   C                     h42-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h42-eth0
mininet> h43 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.2.3                ether   fa:d8:ac:cc:84:99   C                     h43-eth0
10.10.4.1                ether   c2:9e:11:81:bf:0f   C                     h43-eth0
10.10.1.2                ether   12:6d:5a:c6:1d:90   C                     h43-eth0
10.10.2.2                ether   86:5f:bb:aa:c2:b8   C                     h43-eth0
10.10.1.1                ether   62:44:46:4f:e6:8f   C                     h43-eth0
10.10.4.2                ether   d2:5e:cb:b1:02:99   C                     h43-eth0
10.10.2.1                ether   c2:69:7b:96:0d:37   C                     h43-eth0
10.10.1.3                ether   8e:7c:ba:25:17:32   C                     h43-eth0
```

Check the installed flow entries. Here we only need to use the IP destination address as a match.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=1705.946s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.1.3 actions=output:"s1-eth3"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.3 actions=output:"s1-eth4"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.4.3 actions=output:"s1-eth4"
 cookie=0x0, duration=1705.946s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.1.1 actions=output:"s1-eth1"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.1 actions=output:"s1-eth4"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.4.1 actions=output:"s1-eth4"
 cookie=0x0, duration=1705.946s, table=0, n_packets=15, n_bytes=1470, priority=1,ip,nw_dst=10.10.1.2 actions=output:"s1-eth2"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.2 actions=output:"s1-eth4"
 cookie=0x0, duration=1705.946s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.4.2 actions=output:"s1-eth4"
 cookie=0x0, duration=1969.062s, table=0, n_packets=158, n_bytes=10898, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=1705.959s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.2.3 actions=output:"s2-eth3"
 cookie=0x0, duration=1705.958s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.1.3 actions=output:"s2-eth4"
 cookie=0x0, duration=1705.958s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.3 actions=output:"s2-eth5"
 cookie=0x0, duration=1705.957s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.2.1 actions=output:"s2-eth1"
 cookie=0x0, duration=1705.957s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.1.1 actions=output:"s2-eth4"
 cookie=0x0, duration=1705.957s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.1 actions=output:"s2-eth5"
 cookie=0x0, duration=1705.957s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.2.2 actions=output:"s2-eth2"
 cookie=0x0, duration=1705.956s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.1.2 actions=output:"s2-eth4"
 cookie=0x0, duration=1705.955s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.2 actions=output:"s2-eth5"
 cookie=0x0, duration=1969.056s, table=0, n_packets=270, n_bytes=19210, priority=0 actions=CONTROLLER:65535
```

```
*** s3 ------------------------------------------------------------------------
 cookie=0x0, duration=1705.957s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.3 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.957s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.3 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.956s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.3 actions=output:"s3-eth2"
 cookie=0x0, duration=1705.956s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.1 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.956s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.1 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.955s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.1 actions=output:"s3-eth2"
 cookie=0x0, duration=1705.955s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.2 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.954s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.2 actions=output:"s3-eth1"
 cookie=0x0, duration=1705.954s, table=0, n_packets=12, n_bytes=1176, priority=1,ip,nw_dst=10.10.4.2 actions=output:"s3-eth2"
 cookie=0x0, duration=1969.071s, table=0, n_packets=223, n_bytes=16668, priority=0 actions=CONTROLLER:65535
*** s4 ------------------------------------------------------------------------
 cookie=0x0, duration=1705.956s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.4.3 actions=output:"s4-eth3"
 cookie=0x0, duration=1705.955s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.3 actions=output:"s4-eth4"
 cookie=0x0, duration=1705.955s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.3 actions=output:"s4-eth4"
 cookie=0x0, duration=1705.954s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.4.1 actions=output:"s4-eth1"
 cookie=0x0, duration=1705.954s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.1 actions=output:"s4-eth4"
 cookie=0x0, duration=1705.954s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.1 actions=output:"s4-eth4"
 cookie=0x0, duration=1705.954s, table=0, n_packets=16, n_bytes=1568, priority=1,ip,nw_dst=10.10.4.2 actions=output:"s4-eth2"
 cookie=0x0, duration=1705.954s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.1.2 actions=output:"s4-eth4"
 cookie=0x0, duration=1705.953s, table=0, n_packets=6, n_bytes=588, priority=1,ip,nw_dst=10.10.2.2 actions=output:"s4-eth4"
 cookie=0x0, duration=1969.069s, table=0, n_packets=156, n_bytes=10684, priority=0 actions=CONTROLLER:65535
```

Let's take h11 ping h12 as an example to see how our module works. By checking the log information, we can see the first ping passes through the following stages:

```
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler receive
INFO: Received ARP request in switch 00:00:00:00:00:00:00:01 by port 1
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler forwardMessage
INFO: forward ARP request to switch 00:00:00:00:00:00:00:01 on port 2
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler receive
INFO: Received ARP reply in switch 00:00:00:00:00:00:00:01 by port 2
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler forwardMessage
INFO: forward ARP reply to switch 00:00:00:00:00:00:00:01 on port 1
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler receive
INFO: Received dropped IPv4 packet
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler installStaticEntries
INFO: install flow entries in switch 00:00:00:00:00:00:00:01
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler installStaticEntries
INFO: install flow entries in switch 00:00:00:00:00:00:00:02
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler installStaticEntries
INFO: install flow entries in switch 00:00:00:00:00:00:00:03
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler installStaticEntries
INFO: install flow entries in switch 00:00:00:00:00:00:00:04
```

ARP Request/Reply processed by the controller

Install static flow entries for IP Packets

Step 1	h11 tries to ping h12, but its ARP cache is empty, so it sends out an ARP request which causes a PACKET_IN message to go to the controller. If the controller has not seen the source MAC address of the ARP request, saves it in the internal ARP cache.
Step 2	The controller queries the internal ARP cache for the MAC address of h12 and gets nothing, so it redirects the ARP request to h12.
Step 3	h12 sends the ARP reply which again causes a PACKET_IN message to go to the controller.
Step 4	The controller saves the MAC address of h12 to the internal ARP cache and injects the ARP reply to h12.
Step 5	h11 and h22 cached the MAC address of each other and can ping each other now. The ICMP request from h11 causes a PACKET_IN event in the controller which then installs the static flow entries for IP packets in all switches for further pings.

After h11 pinged all other hosts, the internal ARP cache of the controller has saved the MAC addresses of all hosts and the controller will reply to further ARP requests directly.

```
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler receive
INFO: Received ARP request in switch 00:00:00:00:00:00:00:04 by port 2
Jan 06, 2021 12:58:10 PM net.sdnlab.ex3.task31.ARPHandler sendARPReply
INFO: Destination MAC address exists, directly send ARP reply to switch 00:00:00:00:00:00:00:04 on port 2
```

Task2

Create the mininet by using the command
sudo mn --switch ovsk --controller remote,port=6653 --custom ~/ex3/fattree.py --topo fattree --arp
--arp is a critical command in this task because we only match with the IPV4 packet in our routing implementation. The handling of centralized ARP is demonstrated in the previous task.

```
student@sdnfp04:~$ sudo mn --switch ovsk --controller remote,port=6653 --custom ~/ex3
/fattree.py --topo fattree --arp
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s11 s12 s13 s14 s21 s22 s23 s24
*** Adding links:
(h1, s11) (h2, s11) (h3, s12) (h4, s12) (h5, s21) (h6, s21) (h7, s22) (h8, s22) (s13,
 s1) (s13, s2) (s13, s11) (s13, s12) (s14, s1) (s14, s2) (s14, s11) (s14, s12) (s23,
s1) (s23, s2) (s23, s21) (s23, s22) (s24, s1) (s24, s2) (s24, s21) (s24, s22)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 10 switches
s1 s2 s11 s12 s13 s14 s21 s22 s23 s24 ...
*** Starting CLI:
mininet>
```

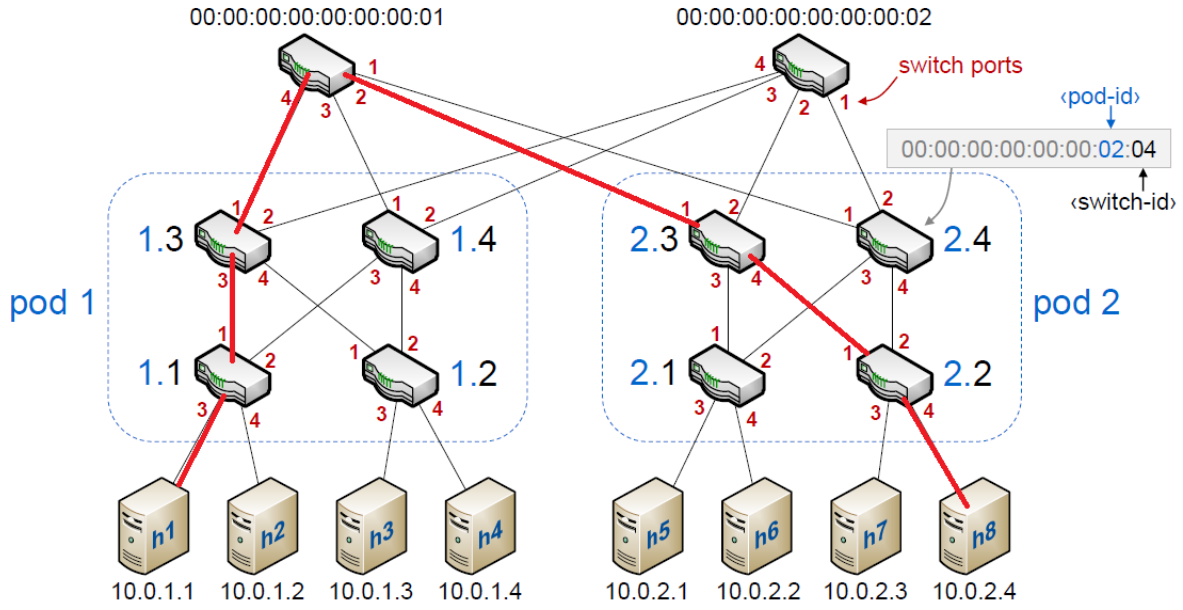
Start the controller by running *Task32.launch.*
To test, let host1 ping host8, `h1 ping -c3 h8`

```
mininet> h1 ping -c3 h8
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=33.0 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.590 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.050 ms
```

The first ping experienced a 33ms delay. This is due to it is forwarded by the controller with PACKET_IN event. The second packet will be forwarded directly by the installed flow.
To find the shortest path, we modified Dijkstra implementation from floodlight in https://github.com/floodlight/floodlight/blob/d737cb05656a6038f4e2277ffb4503d45b7b29cb/src/main/java/net/floodlightcontroller/topology/TopologyInstance.java#L578
In this task, hop counts are used as metrics to find the shortest path.
To examine whether our module correctly finds the shortest path from h1 to h8, we have a look at the newly added flow entries.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=3.669s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s1-eth2"
 cookie=0x0, duration=3.661s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s1-eth4"
 cookie=0x0, duration=33.649s, table=0, n_packets=73, n_bytes=8898, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=33.638s, table=0, n_packets=73, n_bytes=8894, priority=0 actions=CONTROLLER:65535
*** s11 -----------------------------------------------------------------------
 cookie=0x0, duration=33.470s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s11-eth3"
 cookie=0x0, duration=33.470s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.1.2 actions=output:"s11-eth4"
 cookie=0x0, duration=3.675s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s11-eth1"
 cookie=0x0, duration=33.655s, table=0, n_packets=49, n_bytes=5422, priority=0 actions=CONTROLLER:65535
*** s12 -----------------------------------------------------------------------
 cookie=0x0, duration=33.459s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.1.3 actions=output:"s12-eth3"
 cookie=0x0, duration=33.458s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.1.4 actions=output:"s12-eth4"
 cookie=0x0, duration=33.659s, table=0, n_packets=51, n_bytes=5590, priority=0 actions=CONTROLLER:65535
*** s13 -----------------------------------------------------------------------
 cookie=0x0, duration=3.680s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s13-eth1"
 cookie=0x0, duration=3.674s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s13-eth3"
 cookie=0x0, duration=33.660s, table=0, n_packets=75, n_bytes=9078, priority=0 actions=CONTROLLER:65535
*** s14 -----------------------------------------------------------------------
 cookie=0x0, duration=33.682s, table=0, n_packets=74, n_bytes=8984, priority=0 actions=CONTROLLER:65535
*** s21 -----------------------------------------------------------------------
 cookie=0x0, duration=33.472s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.2.1 actions=output:"s21-eth3"
 cookie=0x0, duration=33.471s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.2.2 actions=output:"s21-eth4"
 cookie=0x0, duration=33.668s, table=0, n_packets=53, n_bytes=5766, priority=0 actions=CONTROLLER:65535
*** s22 -----------------------------------------------------------------------
 cookie=0x0, duration=33.485s, table=0, n_packets=0, n_bytes=0, priority=1,ip,nw_dst=10.0.2.3 actions=output:"s22-eth3"
 cookie=0x0, duration=33.485s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s22-eth4"
 cookie=0x0, duration=3.683s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s22-eth1"
 cookie=0x0, duration=33.667s, table=0, n_packets=51, n_bytes=5598, priority=0 actions=CONTROLLER:65535
*** s23 -----------------------------------------------------------------------
 cookie=0x0, duration=3.694s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s23-eth4"
 cookie=0x0, duration=3.685s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s23-eth1"
 cookie=0x0, duration=33.683s, table=0, n_packets=75, n_bytes=9070, priority=0 actions=CONTROLLER:65535
*** s24 -----------------------------------------------------------------------
 cookie=0x0, duration=33.673s, table=0, n_packets=76, n_bytes=9164, priority=0 actions=CONTROLLER:65535
```

The flow entries show the path, which is one of the shortest paths, marked with red lines. One could also see that, all the host-switch are pre-installed at the first of PACKET_IN event. So even though we only performed h1 to h8, all edge-switch (S11, S12, S21, S22) has their hosts flow installed.



We then restart the Mininet and run pingall in the Mininet CLI

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

All pings are received. By this time, in all switches, all the flows of routing are installed. For simplicity, we only show the flow installed on S11. It is shown that all 8 hosts have matching flows.

```
** s11 ------------------------------------------------------------------------
cookie=0x0, duration=880.005s, table=0, n_packets=20, n_bytes=1960, priority=1,ip,nw_dst=10.0.1.1 actions=output:"s11-eth3"
cookie=0x0, duration=880.003s, table=0, n_packets=14, n_bytes=1372, priority=1,ip,nw_dst=10.0.1.2 actions=output:"s11-eth4"
cookie=0x0, duration=837.707s, table=0, n_packets=9, n_bytes=882, priority=1,ip,nw_dst=10.0.2.4 actions=output:"s11-eth1"
cookie=0x0, duration=217.656s, table=0, n_packets=3, n_bytes=294, priority=1,ip,nw_dst=10.0.1.3 actions=output:"s11-eth1"
cookie=0x0, duration=217.642s, table=0, n_packets=3, n_bytes=294, priority=1,ip,nw_dst=10.0.1.4 actions=output:"s11-eth1"
cookie=0x0, duration=217.637s, table=0, n_packets=3, n_bytes=294, priority=1,ip,nw_dst=10.0.2.1 actions=output:"s11-eth1"
cookie=0x0, duration=217.625s, table=0, n_packets=3, n_bytes=294, priority=1,ip,nw_dst=10.0.2.2 actions=output:"s11-eth1"
cookie=0x0, duration=217.616s, table=0, n_packets=3, n_bytes=294, priority=1,ip,nw_dst=10.0.2.3 actions=output:"s11-eth1"
```

Here we briefly explain our control flow.

Control Flow

Our main control is implemented in receivce() callback with PACKET_IN event. In the callback, we first initialize all host-edge switches and install their link on S11, S12, S21, and S22. This part is hard-coded since host-switch link cannot be detected by ILinkDiscoveryService. Then we created the network graph with ILinkDiscoveryService.getSwitchLinks(). The returned switch-link map is equivalent to the adjacency list of a graph of the current topology. The size of switch-link map is 10, which is the number of all the switch in topology.

```
INFO: Get all DatapathID
Jan 06, 2021 7:32:37 PM net.sdnlab.ex3.task32.Reactive createNetworkGraph
INFO: Total switch number: 10
Jan 06, 2021 7:32:37 PM net.sdnlab.ex3.task32.Reactive createNetworkGraph
INFO: Getting switch-link map
Jan 06, 2021 7:32:37 PM net.sdnlab.ex3.task32.Reactive createNetworkGraph
INFO: Current switch links map size: 10
```

This gives a map with switch ID as the key and all the links that end with the current switch as a set as the value. The initialization part is only executed once since the topology is static. We then examine the ethernet packet type with its edge switch (srcDpid). Because the Dijkstra algorithm calculates the shortest path from its src node, to make our program more efficient, we will store the result from dijkstra, which is a minimum spanning tree, in object BroadcastTree. This is already implemented in floodlight in https://github.com/floodlight/floodlight/blob/d737cb05656a6038f4e2277ffb4503d45b7b29cb/src/main/java/net/floodlightcontroller/routing/BroadcastTree.java#L25.
In this way, the next time the same switch with a different destination can use this calculated BroadcastTree to find the shortest path directly, without performing another shortest path.

With the BroadcastTree, We then use `findFlow(srcDpid, dstDpid)` and `installedFlow(Flows)` to find the shortest and then installed. One thing worth noticing here is that the object Link is directional. In the internet network, the links are usually assumed to be bidirectional. In `findFlow(srcDpid, dstDpid)`, it starts with a destination switch and traces by the link map in BroadcastTree to the source. In the end, we used the `injectMessage(eth)` to handle the packet forwarding.

```
receive( ) {
    if (!isGraphCreated) {
      initHostEdgeSwitch();
      installEdgeSwitchToHost();
      createNetworkGraph();
      isGraphCreated = true;
   }
   switch() {
       if (ethIsIPV4Packet) {
           if (!srcSwitchCalculated) {
                calculateShortestPath(srcDpid)
            }
            Flows = findFlow(srcDpid, dstDpid);
            installedFlow(Flows);
            injectMessage(eth);
       }
   }
}
```

```
calculateShortestPath(srcDpid) {
     linkCostMap = initLinkCostMap(HOPCOUNT, switchLinkMap);
     BroadcastTree broadcastTree = dijkstra(srcDpid,
switchLinkMap, linkCostMap);
}
```

Method	Description
`initHostEdgeSwitch()`	initialized the hard-coded edge-switch map for later usage
`installEdgeSwitchToHost()`	install the edge-switch link map on all edge switches
`createNetworkGraph()`	using ILinkDiscoveryService to get switch-link map
`calculateShortestPath()`	calculate the shortest path with link cost map initialization and Dijkstra
`initLinkCostMap(HOPCOUNT, switchLinkMap)`	calculated the link cost by the given map
`Dijkstra(srcDpid, switchLinkMap, linkCostMap)`	calculated the shortest path with the given source switch. It returns with a minimum spanning tree
`findFlow(srcDpid,`	find the route for flow with the given minimum

`dstDpid)`	spanning tree
`installedFlow(Flows)`	create and install all the givens flows on the corresponding switches

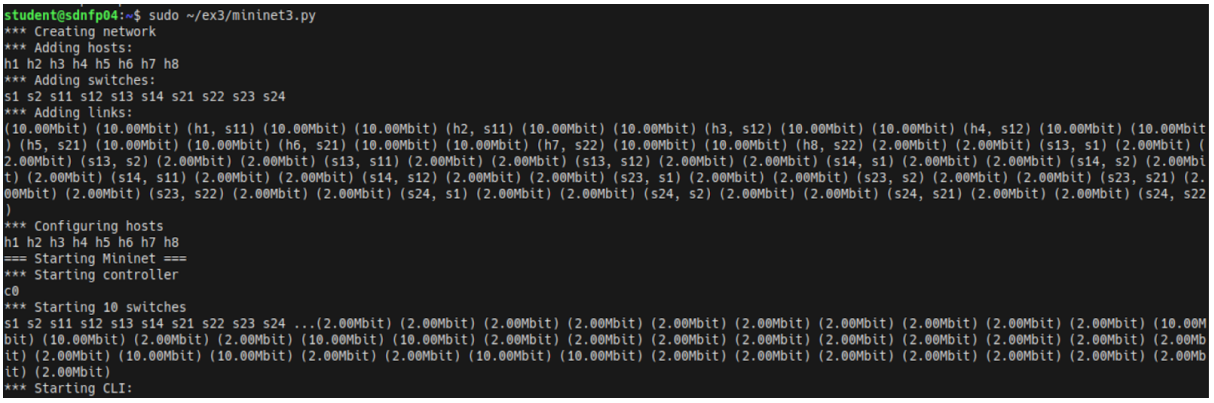
Task3

Reduce statistics collection interval from 10 seconds to 2 seconds in *task33 .properties* file for this task .

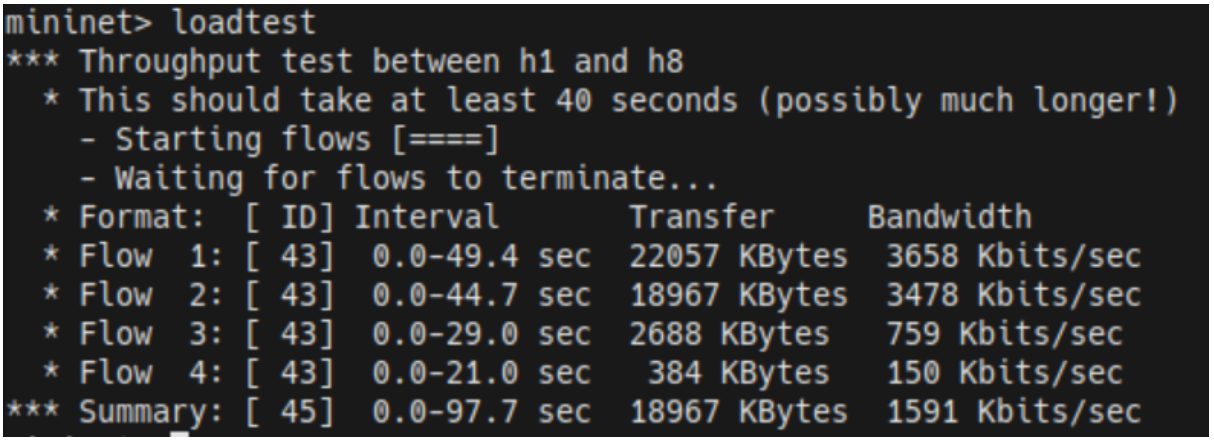
```
net.floodlightcontroller.statistics.StatisticsCollector.collectionIntervalPortStatsSeconds=2
```

Start the controller by running *Task33.launch.*

Start the Mininet script. This loads the fat tree topology with bandwidth limits (max. 10Mbit/s on edge links and max. 2Mbit/s on core links).



Run `loadtest` in the Mininet CLI, which measures the throughput on four connections between two hosts (h1 and h8), waiting 2 seconds between the start of new flows.



Each time the controller receives a TCP flow, it calculates the shortest-path tree based on bandwidth metrics, then finds the path we need and installs flow entries along the path.

```
if (ipv4Pkt.getProtocol() == IpProtocol.TCP) {
      TCP tcp = (TCP) ipv4Pkt.getPayload();
      TransportPort srcPort = tcp.getSourcePort();
      TransportPort dstPort = tcp.getDestinationPort();
```

```
    //calculate the shortest-path tree with link cost map
initialization and Dijkstra
      BroadcastTree broadcastTree = calShortestPath(srcDpid,
BANDWIDTH);

      // find the required path
      List<Link> linkList = findFlow(srcDpid, dstDpid, dstPort,
broadcastTree);

      // install entries in switches along the path
      installFlow(srcAddr, srcPort, dstAddr, dstPort, linkList);
}
```

The bandwidth estimation is implemented in `initLinkCostMap()` method. Here we convert the transmission speed unit from bit per second to byte per second. The Transmission speed is queried using IStatisticsService. The service is binded in `init()`, enabled in `startUp()`. Its dependency is set in `getModuleDependencies()`.

```
case BANDWIDTH:
      logger.info("Using bandwidth for metrics");
      for (NodePortTuple npt : links.keySet()) {
            if (links.get(npt) == null) {
                  continue;
            }
            SwitchPortBandwidth spb =
stats.getBandwidthConsumption(npt.getNodeId(), npt.getPortId());

            long bpsTx = 0;
            if (spb != null) {
                  bpsTx = spb.getBitsPerSecondTx().getValue();
            }
            for (Link link : links.get(npt)) {
                  if (link == null) {
                        continue;
                  }
                  // ensure link weight always larger than zero
                  // cost unit: byte per second
                  int cost = (int) (bpsTx)/8 + 1;
                  // logger.info("link: " + link.toString() + "
bpsTx: " + bpsTx + " cost: " + cost);
                  linkCost.put(link, cost);
            }
      }
      return linkCost;
```

Debug information shows that 8 different flow entries were installed on switches, this is because we have 4 TCP connections.
Note: these entries were installed in a reverse order along the path.
flow 1: s22 - s23 - s1 - s13 - s11 (50080 -> 5001)

Initially only link discovery packets transmitted between the switches, so the bandwidth consumed on each switch can be ignored and therefore, the edge weight was 1.

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.2.4 dstIP: 10.0.1.1 srcPort: 50080 dstPort: 5001 ] ===
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:01:01
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:03
                    link [ srcDpid: 00:00:00:00:00:00:01:03 dstDpid: 00:00:00:00:00:00:01:01]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:01
                    link [ srcDpid: 00:00:00:00:00:00:00:01 dstDpid: 00:00:00:00:00:00:01:03]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:03
                    link [ srcDpid: 00:00:00:00:00:00:02:03 dstDpid: 00:00:00:00:00:00:00:01]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:02
                    link [ srcDpid: 00:00:00:00:00:00:02:02 dstDpid: 00:00:00:00:00:00:02:03]
                    edge weight: 1
```

flow 2: s11 - s13 - s1 - s23 - s22 (5001 -> 50080)

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.1.1 dstIP: 10.0.2.4 srcPort: 5001 dstPort: 50080 ] ===
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:02:02
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:03
                    link [ srcDpid: 00:00:00:00:00:00:02:03 dstDpid: 00:00:00:00:00:00:02:02]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:01
                    link [ srcDpid: 00:00:00:00:00:00:00:01 dstDpid: 00:00:00:00:00:00:02:03]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:03
                    link [ srcDpid: 00:00:00:00:00:00:01:03 dstDpid: 00:00:00:00:00:00:00:01]
                    edge weight: 1
Jan 07, 2021 2:31:59 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:01
                    link [ srcDpid: 00:00:00:00:00:00:01:01 dstDpid: 00:00:00:00:00:00:01:03]
                    edge weight: 1
```

flow 3: s22 - s24 - s1 - s14 - s11 (47926 -> 5001)

The statistics collection interval is 2 seconds and the second TCP flow comes 2 seconds later than the first one, so we get new statistics of the bandwidth consumption and a different route.

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.2.4 dstIP: 10.0.1.1 srcPort: 47926 dstPort: 5001 ] ===
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:01:01
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:04
                    link [ srcDpid: 00:00:00:00:00:00:01:04 dstDpid: 00:00:00:00:00:00:01:01]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:01
                    link [ srcDpid: 00:00:00:00:00:00:00:01 dstDpid: 00:00:00:00:00:00:01:04]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:04
                    link [ srcDpid: 00:00:00:00:00:00:02:04 dstDpid: 00:00:00:00:00:00:00:01]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:02
                    link [ srcDpid: 00:00:00:00:00:00:02:02 dstDpid: 00:00:00:00:00:00:02:04]
                    edge weight: 1
```

flow 4: s11 - s14 - s1 - s24 - s22 (5001 -> 47926)

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.1.1 dstIP: 10.0.2.4 srcPort: 5001 dstPort: 47926 ] ===
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:02:02
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:04
                    link [ srcDpid: 00:00:00:00:00:00:02:04 dstDpid: 00:00:00:00:00:00:02:02]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:01
                    link [ srcDpid: 00:00:00:00:00:00:00:01 dstDpid: 00:00:00:00:00:00:02:04]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:04
                    link [ srcDpid: 00:00:00:00:00:00:01:04 dstDpid: 00:00:00:00:00:00:00:01]
                    edge weight: 1
Jan 07, 2021 2:32:04 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:01
                    link [ srcDpid: 00:00:00:00:00:00:01:01 dstDpid: 00:00:00:00:00:00:01:04]
                    edge weight: 1
```

flow 5: s22 - s24 - s2 - s14 - s11 (45484 -> 5001)

Also, we get new statistics of the bandwidth consumption and a different route for the third TCP flow.

When collecting the statistics, since the second TCP flow is transmitting on the link s22 - s24 link s14 - s11, the edge weights of these two links are quite heavy.

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.2.4 dstIP: 10.0.1.1 srcPort: 45484 dstPort: 5001 ] ===
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:01:01
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:04
                    link [ srcDpid: 00:00:00:00:00:00:01:04 dstDpid: 00:00:00:00:00:00:01:01]
                    edge weight: 10693
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:02
                    link [ srcDpid: 00:00:00:00:00:00:00:02 dstDpid: 00:00:00:00:00:00:01:04]
                    edge weight: 1
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:04
                    link [ srcDpid: 00:00:00:00:00:00:02:04 dstDpid: 00:00:00:00:00:00:00:02]
                    edge weight: 1
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:02
                    link [ srcDpid: 00:00:00:00:00:00:02:02 dstDpid: 00:00:00:00:00:00:02:04]
                    edge weight: 10627
```

flow 6: s11 - s14 - s2 - s24 - s22 (5001 -> 45484)

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.1.1 dstIP: 10.0.2.4 srcPort: 5001 dstPort: 45484 ] ===
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:02:02
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:04
                    link [ srcDpid: 00:00:00:00:00:00:02:04 dstDpid: 00:00:00:00:00:00:02:02]
                    edge weight: 10627
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:02
                    link [ srcDpid: 00:00:00:00:00:00:00:02 dstDpid: 00:00:00:00:00:00:02:04]
                    edge weight: 1
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:04
                    link [ srcDpid: 00:00:00:00:00:00:01:04 dstDpid: 00:00:00:00:00:00:00:02]
                    edge weight: 1
Jan 07, 2021 2:32:09 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:01
                    link [ srcDpid: 00:00:00:00:00:00:01:01 dstDpid: 00:00:00:00:00:00:01:04]
                    edge weight: 10693
```

flow 7: s22 - s23 - s2 - s13 - s11 (43256 -> 5001)

We get new statistics of the bandwidth consumption and a different route for the fourth TCP flow.

When collecting the statistics, since the first TCP flow is transmitting on the link s22 - s23 and link s14 - s11, the edge weights of these two links are quite heavy.

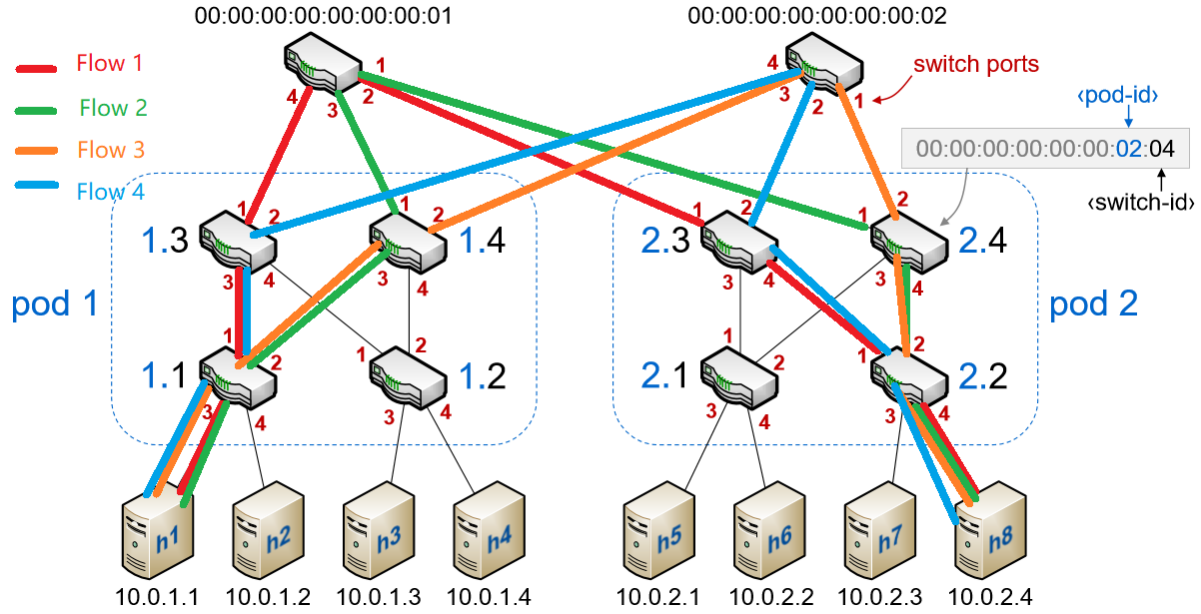
```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.2.4 dstIP: 10.0.1.1 srcPort: 43256 dstPort: 5001 ] ===
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:01:01
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:03
                    link [ srcDpid: 00:00:00:00:00:00:01:03 dstDpid: 00:00:00:00:00:00:01:01]
                    edge weight: 5448
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:02
                    link [ srcDpid: 00:00:00:00:00:00:00:02 dstDpid: 00:00:00:00:00:00:01:03]
                    edge weight: 1
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:03
                    link [ srcDpid: 00:00:00:00:00:00:02:03 dstDpid: 00:00:00:00:00:00:00:02]
                    edge weight: 1
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:02
                    link [ srcDpid: 00:00:00:00:00:00:02:02 dstDpid: 00:00:00:00:00:00:02:03]
                    edge weight: 5413
```

flow 8: s11 - s13 - s2 - s23 - s22 (5001 -> 43256)

```
INFO: === install flows in reverse order: Flow [ srcIP: 10.0.1.1 dstIP: 10.0.2.4 srcPort: 5001 dstPort: 43256 ] ===
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install edge link in switch 00:00:00:00:00:00:02:02
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:02:03
                    link [ srcDpid: 00:00:00:00:00:00:02:03 dstDpid: 00:00:00:00:00:00:02:02]
                    edge weight: 5413
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:00:02
                    link [ srcDpid: 00:00:00:00:00:00:00:02 dstDpid: 00:00:00:00:00:00:02:03]
                    edge weight: 1
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:03
                    link [ srcDpid: 00:00:00:00:00:00:01:03 dstDpid: 00:00:00:00:00:00:00:02]
                    edge weight: 1
Jan 07, 2021 2:32:14 PM net.sdnlab.ex3.task33.Adaptive installFlow
INFO: *** install core link in switch 00:00:00:00:00:00:01:01
                    link [ srcDpid: 00:00:00:00:00:00:01:01 dstDpid: 00:00:00:00:00:00:01:03]
                    edge weight: 5448
```

srcTCPPort	dstTCPPort	sequence of visited switches
50080	5001	s22 - s23 - s1 - s13 - s11
5001	50080	s11 - s13 - s1 - s23 - s22
47926	5001	s22 - s24 - s1 - s14 - s11
5001	47926	s11 - s14 - s1 - s24 - s22
45484	5001	s22 - s24 - s2 - s14 - s11
5001	45484	s11 - s14 - s2 - s24 - s22
43256	5001	s22 - s23 - s2 - s13 - s11
5001	43256	s11 - s13 - s2 - s23 - s22

By visualizing the routes, we can see 4 TCP flows between h1 and h8 have been routed over different paths.



Output the flow entries installed on each switch.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=441.341s, table=0, n_packets=8560, n_bytes=23150864, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=50080,tp_dst=5001 actions=out
put:"s1-eth4"
 cookie=0x0, duration=441.327s, table=0, n_packets=8376, n_bytes=625800, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=50080 actions=outpu
t:"s1-eth2"
 cookie=0x0, duration=436.376s, table=0, n_packets=8035, n_bytes=19952334, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=47926,tp_dst=5001 actions=out
put:"s1-eth3"
 cookie=0x0, duration=436.369s, table=0, n_packets=7900, n_bytes=642268, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=47926 actions=outpu
t:"s1-eth1"
 cookie=0x0, duration=454.714s, table=0, n_packets=200, n_bytes=18369, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=431.375s, table=0, n_packets=1105, n_bytes=89622, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=45484 actions=output
:"s2-eth1"
 cookie=0x0, duration=431.362s, table=0, n_packets=1102, n_bytes=2823796, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=45484,tp_dst=5001 actions=outp
ut:"s2-eth3"
 cookie=0x0, duration=426.369s, table=0, n_packets=184, n_bytes=13744, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=43256 actions=output:
"s2-eth2"
 cookie=0x0, duration=426.369s, table=0, n_packets=184, n_bytes=411152, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=43256,tp_dst=5001 actions=output
:"s2-eth4"
 cookie=0x0, duration=454.718s, table=0, n_packets=200, n_bytes=18358, priority=0 actions=CONTROLLER:65535
```

```
*** s11 -----------------------------------------------------------------------
 cookie=0x0, duration=441.347s, table=0, n_packets=8561, n_bytes=23150930, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=50080,tp_dst=5001 actions=out
put:"s11-eth3"
 cookie=0x0, duration=441.330s, table=0, n_packets=8378, n_bytes=625932, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=50080 actions=outpu
t:"s11-eth1"
 cookie=0x0, duration=436.387s, table=0, n_packets=8035, n_bytes=19952334, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=47926,tp_dst=5001 actions=out
put:"s11-eth3"
 cookie=0x0, duration=436.374s, table=0, n_packets=7900, n_bytes=642268, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=47926 actions=outpu
t:"s11-eth2"
 cookie=0x0, duration=431.376s, table=0, n_packets=1105, n_bytes=89622, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=45484 actions=output
:"s11-eth2"
 cookie=0x0, duration=431.365s, table=0, n_packets=1102, n_bytes=2823796, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=45484,tp_dst=5001 actions=outp
ut:"s11-eth3"
 cookie=0x0, duration=426.372s, table=0, n_packets=184, n_bytes=13744, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=43256 actions=output:
"s11-eth1"
 cookie=0x0, duration=426.371s, table=0, n_packets=184, n_bytes=411152, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=43256,tp_dst=5001 actions=output
:"s11-eth3"
 cookie=0x0, duration=454.718s, table=0, n_packets=120, n_bytes=10660, priority=0 actions=CONTROLLER:65535
*** s12 -----------------------------------------------------------------------
 cookie=0x0, duration=454.725s, table=0, n_packets=115, n_bytes=10290, priority=0 actions=CONTROLLER:65535
```

```
*** s13 -----------------------------------------------------------------------
 cookie=0x0, duration=441.354s, table=0, n_packets=8560, n_bytes=23150864, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=50080,tp_dst=5001 actions=out
put:"s13-eth3"
 cookie=0x0, duration=441.340s, table=0, n_packets=8376, n_bytes=625800, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=50080 actions=outpu
t:"s13-eth1"
 cookie=0x0, duration=426.381s, table=0, n_packets=184, n_bytes=13744, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=43256 actions=output:
"s13-eth2"
 cookie=0x0, duration=426.379s, table=0, n_packets=184, n_bytes=411152, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=43256,tp_dst=5001 actions=output
:"s13-eth3"
 cookie=0x0, duration=454.724s, table=0, n_packets=202, n_bytes=18501, priority=0 actions=CONTROLLER:65535
*** s14 -----------------------------------------------------------------------
 cookie=0x0, duration=436.398s, table=0, n_packets=8035, n_bytes=19952334, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=47926,tp_dst=5001 actions=out
put:"s14-eth3"
 cookie=0x0, duration=436.387s, table=0, n_packets=7900, n_bytes=642268, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=47926 actions=outpu
t:"s14-eth1"
 cookie=0x0, duration=431.388s, table=0, n_packets=1105, n_bytes=89622, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=45484 actions=output
:"s14-eth2"
 cookie=0x0, duration=431.376s, table=0, n_packets=1102, n_bytes=2823796, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=45484,tp_dst=5001 actions=outp
ut:"s14-eth3"
 cookie=0x0, duration=454.730s, table=0, n_packets=202, n_bytes=18517, priority=0 actions=CONTROLLER:65535
```

```
*** s21 ----------------------------------------------------------------------
 cookie=0x0, duration=454.736s, table=0, n_packets=117, n_bytes=10444, priority=0 actions=CONTROLLER:65535
*** s22 ----------------------------------------------------------------------
 cookie=0x0, duration=441.362s, table=0, n_packets=9233, n_bytes=24903922, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=50080,tp_dst=5001 actions=out
put:"s22-eth1"
 cookie=0x0, duration=441.351s, table=0, n_packets=8376, n_bytes=625800, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=50080 actions=outpu
t:"s22-eth4"
 cookie=0x0, duration=436.399s, table=0, n_packets=8733, n_bytes=21694010, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=47926,tp_dst=5001 actions=out
put:"s22-eth2"
 cookie=0x0, duration=436.394s, table=0, n_packets=7900, n_bytes=642268, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=47926 actions=outpu
t:"s22-eth4"
 cookie=0x0, duration=431.395s, table=0, n_packets=1105, n_bytes=89622, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=45484 actions=output
:"s22-eth4"
 cookie=0x0, duration=431.381s, table=0, n_packets=1254, n_bytes=3191348, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=45484,tp_dst=5001 actions=outp
ut:"s22-eth2"
 cookie=0x0, duration=426.391s, table=0, n_packets=184, n_bytes=13744, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=43256 actions=output:
"s22-eth4"
 cookie=0x0, duration=426.386s, table=0, n_packets=205, n_bytes=459818, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=43256,tp_dst=5001 actions=output
:"s22-eth1"
 cookie=0x0, duration=454.733s, table=0, n_packets=122, n_bytes=10810, priority=0 actions=CONTROLLER:65535
```

```
*** s23 ----------------------------------------------------------------------
 cookie=0x0, duration=441.367s, table=0, n_packets=8560, n_bytes=23150864, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=50080,tp_dst=5001 actions=out
put:"s23-eth1"
 cookie=0x0, duration=441.354s, table=0, n_packets=8376, n_bytes=625800, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=50080 actions=outpu
t:"s23-eth4"
 cookie=0x0, duration=426.393s, table=0, n_packets=184, n_bytes=13744, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=43256 actions=output:
"s23-eth4"
 cookie=0x0, duration=426.393s, table=0, n_packets=184, n_bytes=411152, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=43256,tp_dst=5001 actions=output
:"s23-eth2"
 cookie=0x0, duration=454.739s, table=0, n_packets=198, n_bytes=18234, priority=0 actions=CONTROLLER:65535
*** s24 ----------------------------------------------------------------------
 cookie=0x0, duration=436.406s, table=0, n_packets=8035, n_bytes=19952334, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=47926,tp_dst=5001 actions=out
put:"s24-eth1"
 cookie=0x0, duration=436.402s, table=0, n_packets=7900, n_bytes=642268, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=47926 actions=outpu
t:"s24-eth4"
 cookie=0x0, duration=431.401s, table=0, n_packets=1105, n_bytes=89622, priority=1,tcp,nw_src=10.0.1.1,nw_dst=10.0.2.4,tp_src=5001,tp_dst=45484 actions=output
:"s24-eth4"
 cookie=0x0, duration=431.390s, table=0, n_packets=1102, n_bytes=2823796, priority=1,tcp,nw_src=10.0.2.4,nw_dst=10.0.1.1,tp_src=45484,tp_dst=5001 actions=outp
ut:"s24-eth2"
 cookie=0x0, duration=454.743s, table=0, n_packets=197, n_bytes=18166, priority=0 actions=CONTROLLER:65535
```

Comparison

Start the module Reactive by running *Task32.launch.*
Start the Mininet script and run `loadtest` in the Mininet CLI.

```
mininet> loadtest
*** Throughput test between h1 and h8
  * This should take at least 40 seconds (possibly much longer!)
    - Starting flows [====]
    - Waiting for flows to terminate...
  * Format:   [ ID] Interval       Transfer     Bandwidth
  * Flow  1: [ 43]  0.0-47.1 sec  17680 KBytes  3074 Kbits/sec
  * Flow  2: [ 43]  0.0-32.4 sec  2304 KBytes   582 Kbits/sec
  * Flow  3: [ 43]  0.0-25.4 sec   512 KBytes   165 Kbits/sec
  * Flow  4: [ 43]  0.0-20.4 sec  86.3 KBytes  34.7 Kbits/sec
*** Summary: [ 47]  0.0-88.1 sec  86.3 KBytes  8.02 Kbits/sec
```

`loadtest` result from Task32.

```
mininet> loadtest
*** Throughput test between h1 and h8
  * This should take at least 40 seconds (possibly much longer!)
    - Starting flows [====]
    - Waiting for flows to terminate...
  * Format:   [ ID] Interval       Transfer     Bandwidth
  * Flow  1: [ 43]  0.0-49.4 sec  22057 KBytes  3658 Kbits/sec
  * Flow  2: [ 43]  0.0-44.7 sec  18967 KBytes  3478 Kbits/sec
  * Flow  3: [ 43]  0.0-29.0 sec  2688 KBytes   759 Kbits/sec
  * Flow  4: [ 43]  0.0-21.0 sec   384 KBytes   150 Kbits/sec
*** Summary: [ 45]  0.0-97.7 sec  18967 KBytes  1591 Kbits/sec
```

By comparing both solutions, clearly the overall throughput of Task33 is higher than that of Task32.
Since Task32 uses hopcount as metrics, we always get the same shortest-path tree for a given source node and all TCP flows will be routed over the same path. So the path is crowded and the throughput decreases prominently from the first TCP flow to the fourth one.
Now in Task33, bandwidth is used as metrics, so we can get different shortest-path trees with the change of bandwidth and therefore, the load can be distributed evenly over links. Because we get non-overlapping routes for the first two TCP flows, so they have a similar throughput which is quite high, and following TCP flows use some links which have been occupied, so the throughput will decrease. But the overall throughput is still higher than that of Task32.

Reference

- iperf: https://linux.die.net/man/1/iperf
- Floodlight Services: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/40402986/Floodlight+Services
- BroadcastTree (Dijkstra return type): https://github.com/floodlight/floodlight/blob/d737cb05656a6038f4e2277ffb4503d45b7b29cb/src/main/java/net/floodlightcontroller/routing/BroadcastTree.java#L25
- Link: http://floodlight.github.io/floodlight/javadoc/floodlight/net/floodlightcontroller/linkdiscovery/Link.html
- Graph and its representations: https://www.geeksforgeeks.org/graph-and-its-representations/