Software Defined Networking
Assignment 01

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

Task 1

Fire up a terminal and start Mininet with an external controller:

```
student@sdnfp04:~$ sudo mn --switch ovsk --controller remote,port=6653
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Send three pings from host 1 to host 2:

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2055ms
pipe 3
```

Output the current flows over switch s1:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
```

Were the pings successful?
No, the packets were dropped because the controller was not started.

Which flows are installed in the switch?
No flow table entries are installed in the switch, because the controller was not started and initially the flow table of s1 is empty.

In a new terminal, start the Floodlight controller. By default, Floodlight loads the Forwarding module which does reactive entry pushing.

```
student@sdnfp04:~$ /opt/floodlight/floodlight-default.sh
```

When the controller has been started completely, go back to the Mininet CLI and resend three pings from host 1 to host 2:

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=112 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.189 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.036 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.036/37.571/112.490/52.975 ms
```

Output the flows of s1:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x20000001000000, duration=6.070s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, priority=1,ip,in_port="s1-eth1",dl_src=96:c3:75:98:e4:d1,dl_dst=
fa:ae:75:0a:42:a7,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:"s1-eth2"
 cookie=0x20000002000000, duration=6.058s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, priority=1,ip,in_port="s1-eth2",dl_src=fa:ae:75:0a:42:a7,dl_dst=
96:c3:75:98:e4:d1,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x20000003000000, duration=0.942s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port="s1-eth2",dl_src=fa:ae:75:0a:42:a7,dl_dst=9
6:c3:75:98:e4:d1 actions=output:"s1-eth1"
 cookie=0x20000004000000, duration=0.940s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port="s1-eth1",dl_src=96:c3:75:98:e4:d1,dl_dst=f
a:ae:75:0a:42:a7 actions=output:"s1-eth2"
 cookie=0x0, duration=13.529s, table=0, n_packets=6, n_bytes=364, priority=0 actions=CONTROLLER:65535
```

How has the output changed?

All pings became successful and four flow entries were installed in the switch.

What is the reason for this change?

Initially h1 did not know the MAC address of h2, so h1 sent an ARP Request to h2, which caused a packet_in event at the controller. The controller then sent a packet_out message to flood the broadcast packet to other ports on the switch. When h2 received the ARP Request, it would return an ARP Reply. This reply would also be redirected to the controller, which then sent it to h1 and pushed down a flow entry. As shown in the output of command `dpctl dump-flows`, an ARP Request from h1 would be forwarded to h2 and the ARP Reply from h2 would be forwarded to h1.

When h1 knew the MAC address of h2, it sent an ICMP Request to h2, which again caused a packet_in event at the controller and a new flow entry was added to the switch. The process for ICMP Reply was the same, so two entries were installed, which greatly reduced the ping time of following pings.

Task 2

Steps:

Start the floodlight controller with noforwarding configuration, which means the controller cannot do reactive entry pushing.

```
student@sdnfp04:~$ /opt/floodlight/floodlight-noforwarding.sh
```

Create a Mininet network with custom topology task12topo, where `--mac` option automatically sets host MACs which are easy to read, `--arp` option sets all-pairs ARP entries.

```
student@sdnfp04:~$ sudo mn --mac --arp --switch ovsk --controller remote,port=6653 --custom ex1/task12topo.py --topo task12topo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
```

ARP caches of all hosts have been filled:

```
mininet> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.2                 ether   00:00:00:00:00:02   CM                    h1-eth0
10.0.0.4                 ether   00:00:00:00:00:04   CM                    h1-eth0
10.0.0.3                 ether   00:00:00:00:00:03   CM                    h1-eth0
mininet> h2 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.3                 ether   00:00:00:00:00:03   CM                    h2-eth0
10.0.0.4                 ether   00:00:00:00:00:04   CM                    h2-eth0
10.0.0.1                 ether   00:00:00:00:00:01   CM                    h2-eth0
mininet> h3 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.1                 ether   00:00:00:00:00:01   CM                    h3-eth0
10.0.0.2                 ether   00:00:00:00:00:02   CM                    h3-eth0
10.0.0.4                 ether   00:00:00:00:00:04   CM                    h3-eth0
mininet> h4 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.2                 ether   00:00:00:00:00:02   CM                    h4-eth0
10.0.0.3                 ether   00:00:00:00:00:03   CM                    h4-eth0
10.0.0.1                 ether   00:00:00:00:00:01   CM                    h4-eth0
```

No flow entries installed in both switches, so all pings failed.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=88.306s, table=0, n_packets=40, n_bytes=3953, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=88.305s, table=0, n_packets=42, n_bytes=4125, priority=0 actions=CONTROLLER:65535
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
```

The Static Entry Pusher module of Floodlight is exposed via a REST API, so we can use HTTP Post request `curl -X POST -d '{<JSON attributes>}' http://localhost:8080/wm/staticentrypusher/json` to insert flow entries into our switches proactively. All flows have been written in the script task12.sh.

```
student@sdnfp04:~$ sh ex1/task12.sh
{"status":"Deleted all flows/groups."}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" :
"Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}student@sdnfp04:~$
```

Now the custom flow entries have been installed in s1 and s2:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0xafffffff32deb34, duration=30.748s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s1-eth2" actions=mod_vlan_vid:1,output:"s1-eth1"
 cookie=0xafffffff3bd423f, duration=30.722s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s1-eth3" actions=mod_vlan_vid:2,output:"s1-eth1"
 cookie=0xa000006edcf58b, duration=30.736s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s1-eth1",dl_vlan=1 actions=strip_vlan,output:"s1-eth2"
 cookie=0xaffffffe501b39c, duration=30.706s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s1-eth1",dl_vlan=2 actions=strip_vlan,output:"s1-eth3"
 cookie=0x0, duration=527.073s, table=0, n_packets=83, n_bytes=7304, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0xa00000471f4d67, duration=30.693s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s2-eth3" actions=mod_vlan_vid:1,output:"s2-eth1"
 cookie=0xa0000047aea472, duration=30.665s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s2-eth2" actions=mod_vlan_vid:2,output:"s2-eth1"
 cookie=0xafffff9ed0e594, duration=30.679s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s2-eth1",dl_vlan=1 actions=strip_vlan,output:"s2-eth3"
 cookie=0xa0000014f5a3a5, duration=30.650s, table=0, n_packets=0, n_bytes=0, priority=65535,in_port="s2-eth1",dl_vlan=2 actions=strip_vlan,output:"s2-eth2"
 cookie=0x0, duration=527.072s, table=0, n_packets=84, n_bytes=7378, priority=0 actions=CONTROLLER:65535
```

VLAN isolation realized:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X
h2 -> X X h4
h3 -> h1 X X
h4 -> X h2 X
*** Results: 66% dropped (4/12 received)
```

Explanation:

In this task, we set up two VLAN groups using VLAN ID for matching, Since ethernet frame from h1 to h3 in VLAN1 and frame from h2 to h4 will all be forwarded by S1, S2, we can use ports of switch to decide whether the hosts belongs to the VLAN without using MAC and IP for matching. On both S1 and S2, we configured VLAN groups with specific ID. The ID is pushed into the ethernet frame and is popped when the frame is delivered to the host so that host doesn't know each other's VLAN ID . This transparency prevents hosts from knowing and hearing from other hosts in different VLAN, which is the same behaviour as hosts in a physical LAN.

Task 3

Steps:

Start the floodlight controller with noforwarding configuration:

```
student@sdnfp04:~$ /opt/floodlight/floodlight-noforwarding.sh
```

Create a Mininet network with custom topology task12topo without `--arp` option, which means the ARP caches of all hosts are empty initially.

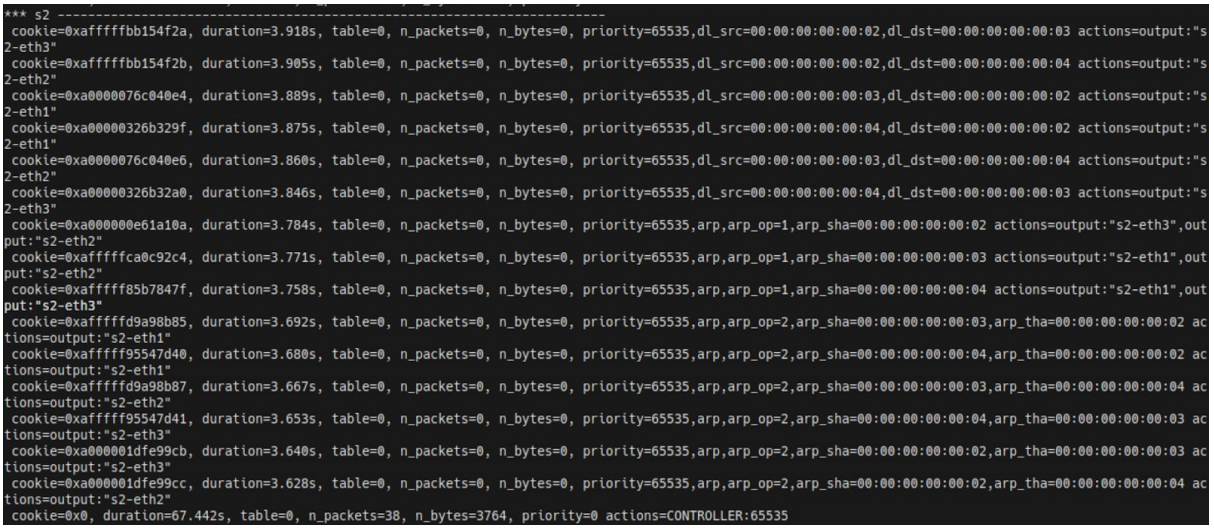
```
student@sdnfp04:~$ sudo mn --mac --switch ovsk --controller remote,port=6653 --custom ex1/task12topo.py --topo task12topo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=44.227s, table=0, n_packets=37, n_bytes=3765, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=44.233s, table=0, n_packets=36, n_bytes=3690, priority=0 actions=CONTROLLER:65535
mininet> h1 arp
mininet> h2 arp
mininet> h3 arp
mininet> h4 arp
```

Install flow entries in s1 and s2

```
student@sdnfp04:~$ sh ex1/task13.sh
{"status":"Deleted all flows/groups."}{"status" : "Entry pushed"}{"status" : "En
try pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "E
ntry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "
Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" :
"Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status" :
 "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status"
: "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status"
 : "Entry pushed"}{"status" : "Entry pushed"}{"status" : "Entry pushed"}{"status
" : "Entry pushed"}{"status" : "Entry pushed"}student@sdnfp04:~$ sh ex1/task13.s
```

Print the flow entries of s1 and s2

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0xaffffff9877b091, duration=6.199s, table=0, n_packets=0, n_bytes=0, priority=65535,ip,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output
:"s1-eth1"
 cookie=0xaffffff9877b092, duration=6.167s, table=0, n_packets=0, n_bytes=0, priority=65535,ip,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04 actions=output
:"s1-eth1"
 cookie=0xa000005422a24b, duration=6.151s, table=0, n_packets=0, n_bytes=0, priority=65535,ip,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0xa000000fcd9406, duration=6.122s, table=0, n_packets=0, n_bytes=0, priority=65535,ip,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0xaffffffe806d1f3, duration=6.011s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=1,arp_sha=00:00:00:00:00:02 actions=output:"s1-eth1"
 cookie=0xaffffffe806d1f4, duration=5.996s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=1,arp_sha=00:00:00:00:00:03 actions=output:"s1-eth3"
 cookie=0xaffffffe806d1f5, duration=5.978s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=1,arp_sha=00:00:00:00:00:04 actions=output:"s1-eth3"
 cookie=0xaffffffb60fb32, duration=5.915s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=2,arp_sha=00:00:00:00:00:02,arp_tha=00:00:00:00:00:03 ac
tions=output:"s1-eth1"
 cookie=0xaffffffb60fb33, duration=5.903s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=2,arp_sha=00:00:00:00:00:02,arp_tha=00:00:00:00:00:04 ac
tions=output:"s1-eth1"
 cookie=0xa0000072b6dea7, duration=5.884s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=2,arp_sha=00:00:00:00:00:04,arp_tha=00:00:00:00:00:02 ac
tions=output:"s1-eth3"
 cookie=0xaffffffb70becec, duration=5.871s, table=0, n_packets=0, n_bytes=0, priority=65535,arp,arp_op=2,arp_sha=00:00:00:00:00:03,arp_tha=00:00:00:00:00:02 ac
tions=output:"s1-eth3"
 cookie=0x0, duration=39.008s, table=0, n_packets=35, n_bytes=3580, priority=0 actions=CONTROLLER:65535
```



Test the connectivity between all hosts. We can see that h2, h3 and h4 have formed a VLAN and cached the MACs of each other.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X h3 h4
h3 -> X h2 h4
h4 -> X h2 h3
*** Results: 50% dropped (6/12 received)
mininet> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.2                         (incomplete)                              h1-eth0
10.0.0.4                         (incomplete)                              h1-eth0
10.0.0.3                         (incomplete)                              h1-eth0
mininet> h2 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.1                         (incomplete)                              h2-eth0
10.0.0.4                 ether   00:00:00:00:00:04   C                     h2-eth0
10.0.0.3                 ether   00:00:00:00:00:03   C                     h2-eth0
mininet> h3 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.4                 ether   00:00:00:00:00:04   C                     h3-eth0
10.0.0.2                 ether   00:00:00:00:00:02   C                     h3-eth0
10.0.0.1                         (incomplete)                              h3-eth0
mininet> h4 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.1                         (incomplete)                              h4-eth0
10.0.0.2                 ether   00:00:00:00:00:02   C                     h4-eth0
10.0.0.3                 ether   00:00:00:00:00:03   C                     h4-eth0
```

Explanation

In this task, we cannot use VLAN ID in the ethernet frame for creating VLAN because hosts h3 and h4 share the same switch. The switch cannot push VLAN from input port and pop VLAN ID to output port at same time. Otherwise, one of the hosts will know the VLAN ID. It is not feasible to use VLAN ID for broadcasting. Therefore, we have to decompose all broadcast flow into several two way flows. Moreover, since the ARP table is empty in the beginning for all hosts, we also need to take care of ARP broadcasts.

Task 4

Start Mininet with our custom topology and use `net` to display links:

```
student@sdnfp04:~$ sudo ~/ex1/task14.py
Unable to contact the remote controller at 127.0.0.1:6653
=== Starting Mininet ===
*** Configuring hosts
h11 h12 h13 h21 h22 h23 h41 h42 h43
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

```
mininet> net
h11 h11-eth0:s1-eth1
h12 h12-eth0:s1-eth2
h13 h13-eth0:s1-eth3
h21 h21-eth0:s2-eth1
h22 h22-eth0:s2-eth2
h23 h23-eth0:s2-eth3
h41 h41-eth0:s4-eth1
h42 h42-eth0:s4-eth2
h43 h43-eth0:s4-eth3
s1 lo:  s1-eth1:h11-eth0 s1-eth2:h12-eth0 s1-eth3:h13-eth0 s1-eth4:s2-eth4
s2 lo:  s2-eth1:h21-eth0 s2-eth2:h22-eth0 s2-eth3:h23-eth0 s2-eth4:s1-eth4 s2-eth5:s3-eth1
s3 lo:  s3-eth1:s2-eth5 s3-eth2:s4-eth4
s4 lo:  s4-eth1:h41-eth0 s4-eth2:h42-eth0 s4-eth3:h43-eth0 s4-eth4:s3-eth2
c0
```

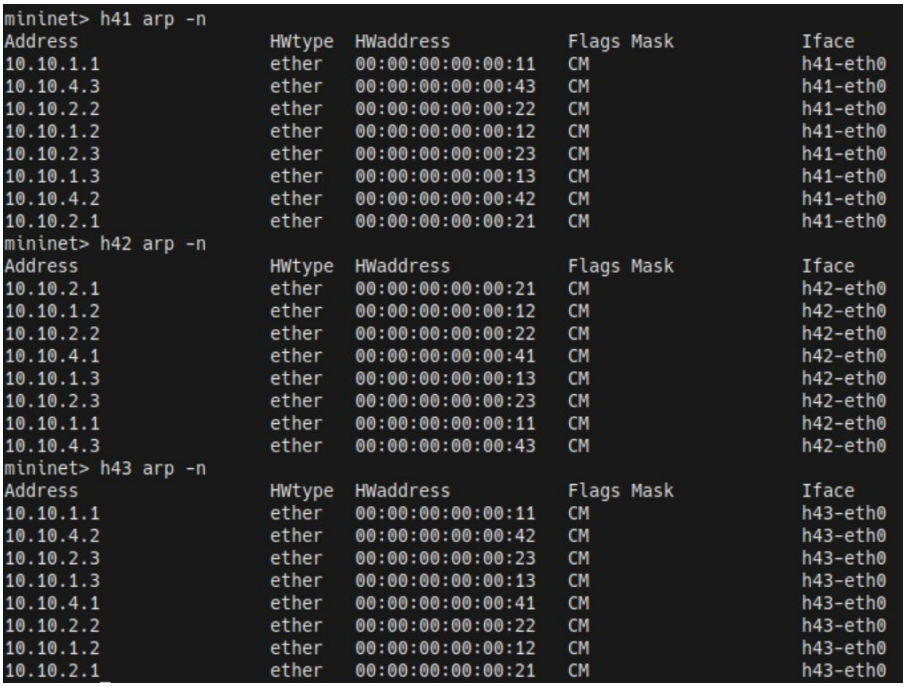
Display the MAC address of each host:

```
mininet> py h11.MAC()
00:00:00:00:00:11
mininet> py h12.MAC()
00:00:00:00:00:12
mininet> py h13.MAC()
00:00:00:00:00:13
mininet> py h21.MAC()
00:00:00:00:00:21
mininet> py h22.MAC()
00:00:00:00:00:22
mininet> py h23.MAC()
00:00:00:00:00:23
mininet> py h41.MAC()
00:00:00:00:00:41
mininet> py h42.MAC()
00:00:00:00:00:42
mininet> py h43.MAC()
00:00:00:00:00:43
```

Display the ARP cache of each host:

```
mininet> h11 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.2                ether   00:00:00:00:00:42   CM                    h11-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h11-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h11-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h11-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h11-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h11-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h11-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h11-eth0
mininet> h12 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.2                ether   00:00:00:00:00:42   CM                    h12-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h12-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h12-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h12-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h12-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h12-eth0
10.10.1.1                ether   00:00:00:00:00:11   CM                    h12-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h12-eth0
mininet> h13 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.1                ether   00:00:00:00:00:11   CM                    h13-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h13-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h13-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h13-eth0
10.10.4.2                ether   00:00:00:00:00:42   CM                    h13-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h13-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h13-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h13-eth0
```

```
mininet> h21 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.1                ether   00:00:00:00:00:11   CM                    h21-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h21-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h21-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h21-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h21-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h21-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h21-eth0
10.10.4.2                ether   00:00:00:00:00:42   CM                    h21-eth0
mininet> h22 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.4.2                ether   00:00:00:00:00:42   CM                    h22-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h22-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h22-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h22-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h22-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h22-eth0
10.10.1.1                ether   00:00:00:00:00:11   CM                    h22-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h22-eth0
mininet> h23 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.3                ether   00:00:00:00:00:13   CM                    h23-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h23-eth0
10.10.4.2                ether   00:00:00:00:00:42   CM                    h23-eth0
10.10.1.1                ether   00:00:00:00:00:11   CM                    h23-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h23-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h23-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h23-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h23-eth0
```



```
mininet> h41 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.1                ether   00:00:00:00:00:11   CM                    h41-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h41-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h41-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h41-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h41-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h41-eth0
10.10.4.2                ether   00:00:00:00:00:42   CM                    h41-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h41-eth0
mininet> h42 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.2.1                ether   00:00:00:00:00:21   CM                    h42-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h42-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h42-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h42-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h42-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h42-eth0
10.10.1.1                ether   00:00:00:00:00:11   CM                    h42-eth0
10.10.4.3                ether   00:00:00:00:00:43   CM                    h42-eth0
mininet> h43 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.1                ether   00:00:00:00:00:11   CM                    h43-eth0
10.10.4.2                ether   00:00:00:00:00:42   CM                    h43-eth0
10.10.2.3                ether   00:00:00:00:00:23   CM                    h43-eth0
10.10.1.3                ether   00:00:00:00:00:13   CM                    h43-eth0
10.10.4.1                ether   00:00:00:00:00:41   CM                    h43-eth0
10.10.2.2                ether   00:00:00:00:00:22   CM                    h43-eth0
10.10.1.2                ether   00:00:00:00:00:12   CM                    h43-eth0
10.10.2.1                ether   00:00:00:00:00:21   CM                    h43-eth0
```

We can see that the ARP cache of each host has been filled statically. “CM” means that the entry is complete and marked permanently.