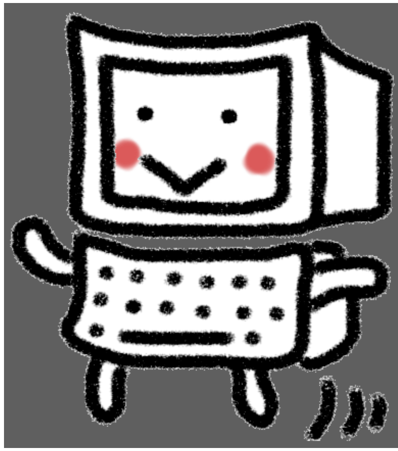


Final Project

SDN-IP

Date: 2020/06/11

Deadline: 2020/07/02

Outline

❑ SDN-IP Introduction

❑ Scenario

❑ Environment Setup

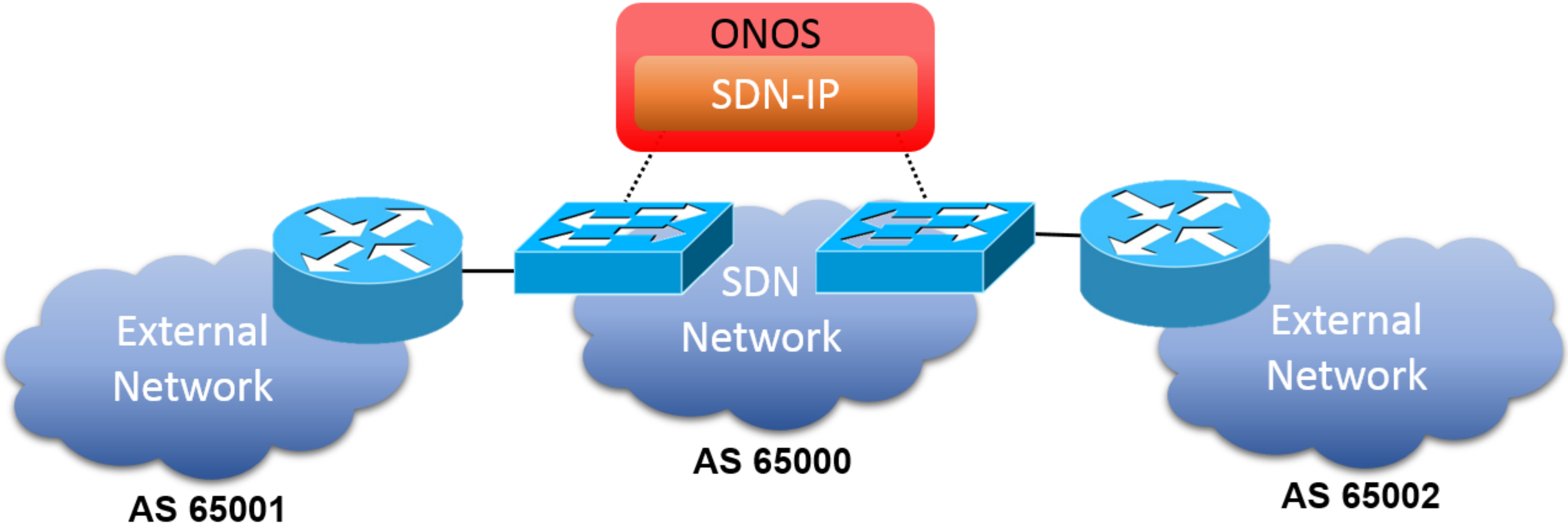
❑ Target Topology

❑ Report Submission

❑ References

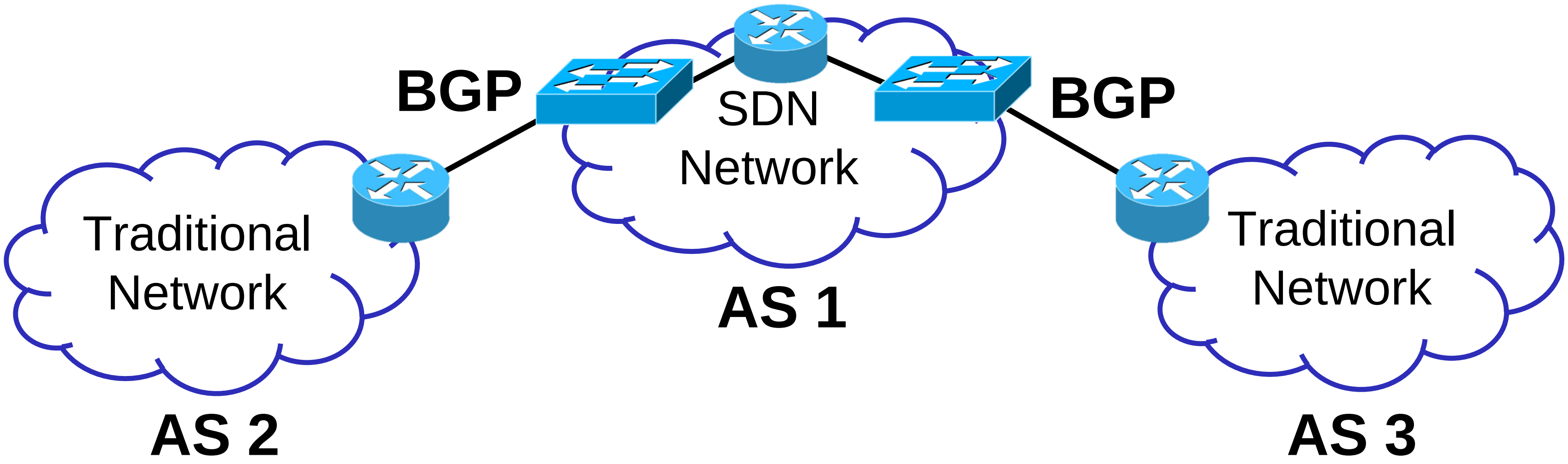
Introduction of SDN-IP

- ❑ SDN-IP is an ONOS application.
- ❑ SDN-IP allows SDN network to connect to External networks.
  - ■ External network can be SDN networks or traditional networks



Scenario

- SDN Network connects to traditional networks

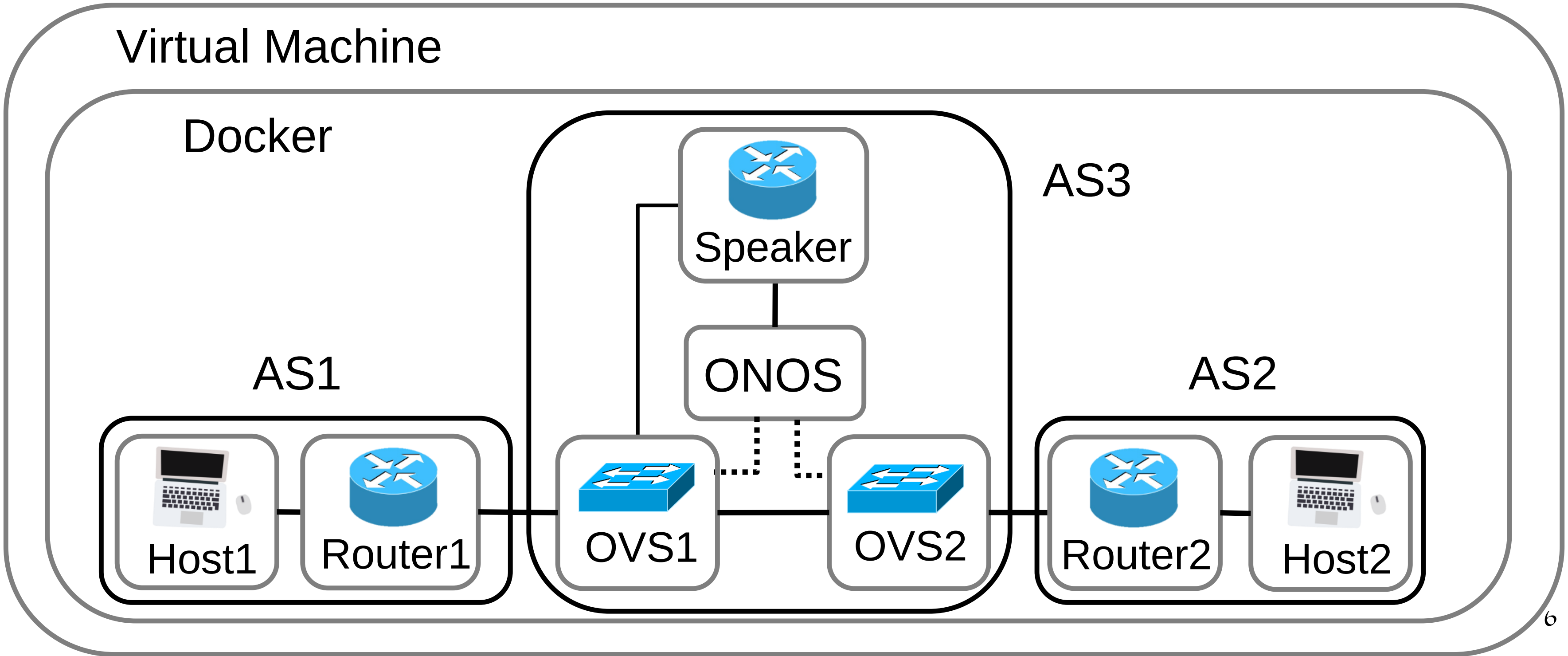


Outline

- SDN-IP Introduction
- Scenario
- Environment Setup
- Target Topology
- Report Submission
- References

Environment

- Virtual Machine
  - Ubuntu 16.04
- Docker images
  - ubuntu:16.04
  - onosproject/onos:2.2.0
  - openshift/openvswitch:latest



Environment Setup

1. Pull docker images
2. Create Containers
3. Setup Container Networks
4. Configure Host Gateways
5. Setup SDN network
6. Activate ONOS Applications
7. Setup OVS
8. Configure Routers
9. Setup ONOS Network Configuration

1. Pull docker images

❑ Docker images

- ubuntu:16.04

```
~$ sudo docker pull ubuntu:16.04
```

- onosproject/onos:2.2.0

```
~$ sudo docker pull onosproject/onos:2.2.0
```

- openshift/openvswitch:latest

```
~$ sudo docker pull openshift/openvswitch:latest
```

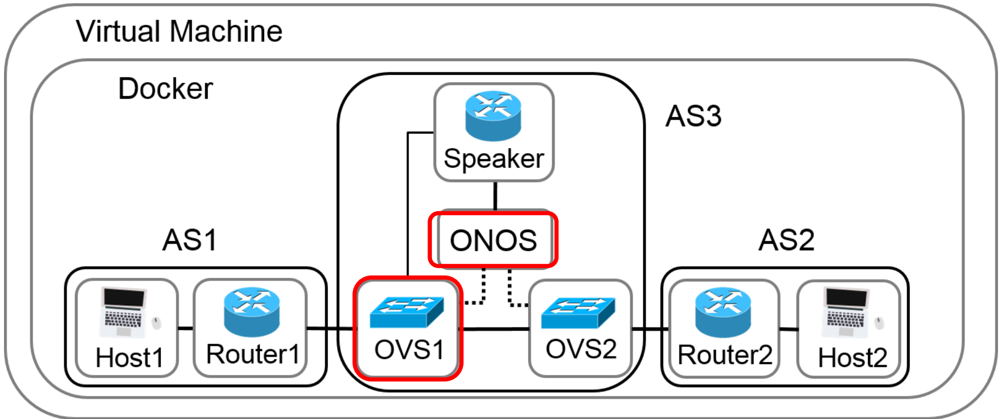
2. Create Containers

- ❑ Create eight containers with specific images
- ❑ E.g., Create container for ONOS

```
~$ sudo docker run --privileged \
--cap-add NET_ADMIN --cap-add NET_BROADCAST \
-d –it --name ONOS onosproject/onos:2.2.0
```

- ❑ E.g., Create container for OVS (ovs1) (ovs2)

```
~$ sudo docker run --privileged \
--cap-add NET_ADMIN --cap-add NET_BROADCAST \
-d –it --name OVS1 openshift/openvswitch:latest
```



3. Setup Container Networks - Host and Router

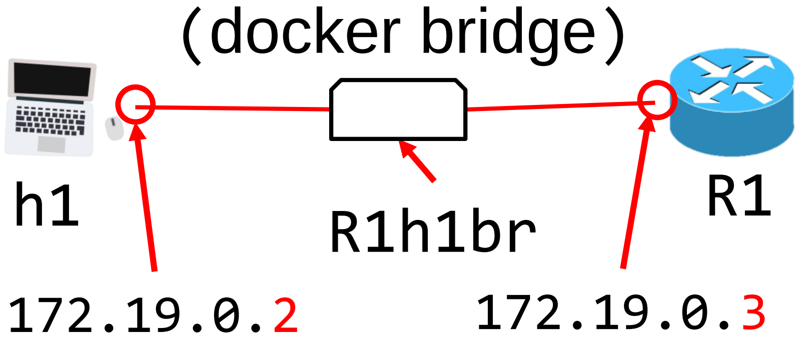
- Assume host (h1) and virtual router (R1) containers created
- Create a bridge for domain serving by R1

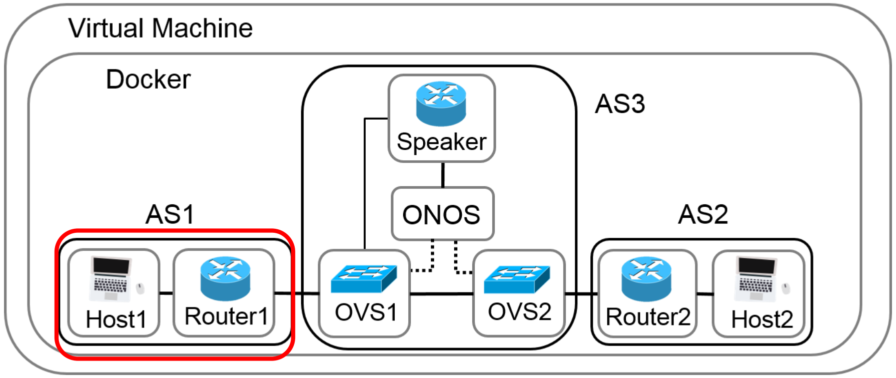
```
~$ sudo docker network create R1h1br
```

  - R1h1br: Bridge name

- Connect containers h1, R1 to bridge R1h1br

```
~$ sudo docker network connect R1h1br R1
~$ sudo docker network connect R1h1br h1
```





- Repeat network setup procedure for each domain

3. Setup Container Networks - Router and OVS

□ Assume virtual router (R1) and OVS (OVS1) containers created

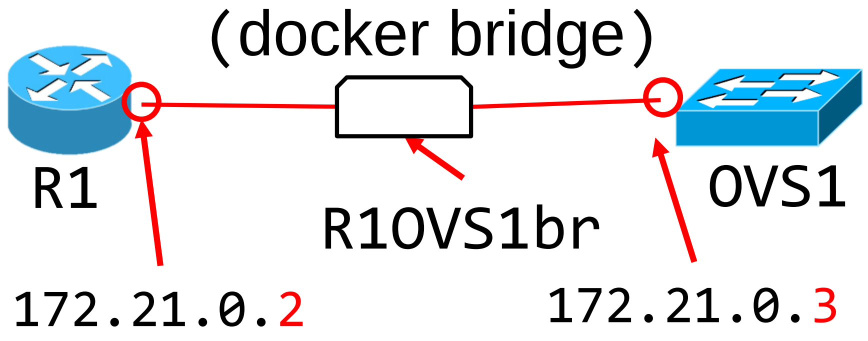
□ Create docker bridge

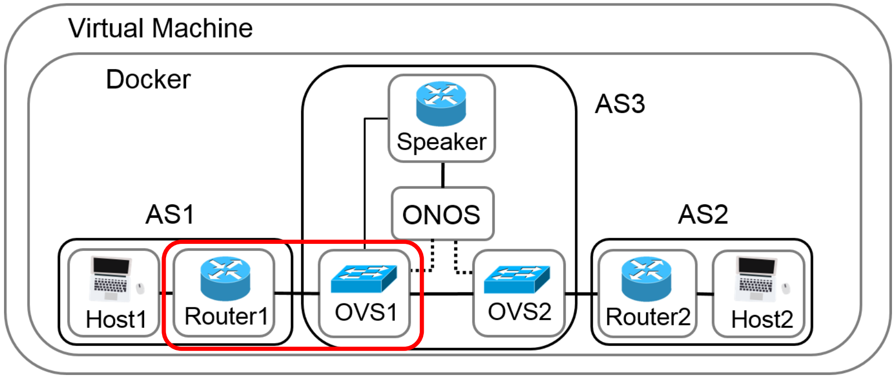
```
~$ sudo docker network create
```

- R1OVS1br: Bridge name

□ Connect containers R1, OVS1 to bridge R1OVS1br

```
~$ sudo docker network connect R1OVS1br R1
~$ sudo docker network connect R1OVS1br OVS1
```





□ Repeat network setup procedure for each domain

3. Setup Container Networks – beween OVSs

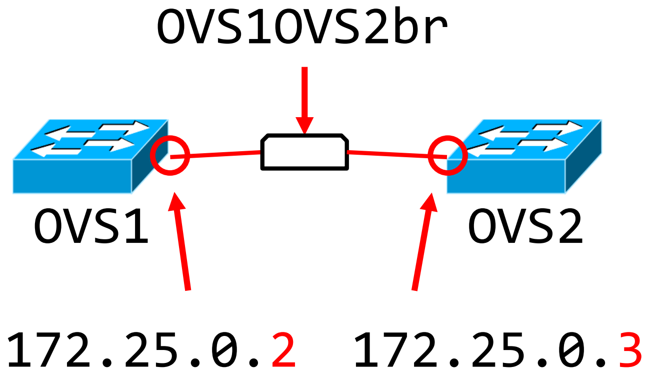
- ❑ Assume OVS1 and OVS2 containers created
- ❑ Create docker bridge

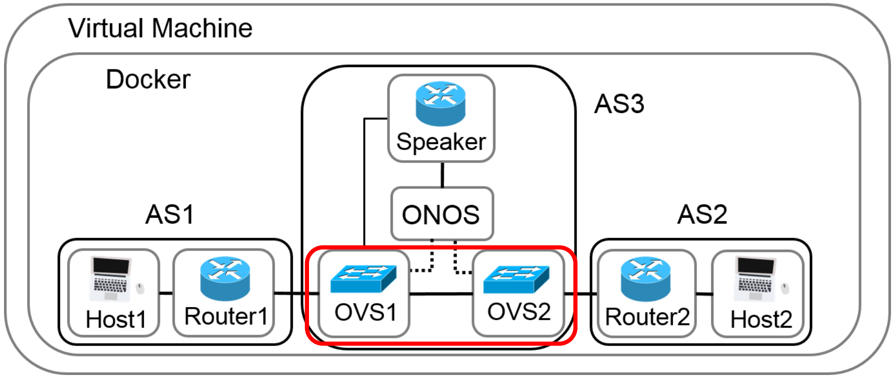
```
~$ sudo docker network create OVS1OVS2br
```

  - ■ OVS1OVS2br: Bridge name

- ❑ Connect containers OVS1, OVS2 to bridge R1OVS1br

```
~$ sudo docker network connect OVS1OVS2br OVS1
~$ sudo docker network connect OVS1OVS2br OVS2
```





3. Setup Container Networks - OVS, Speaker and ONOS

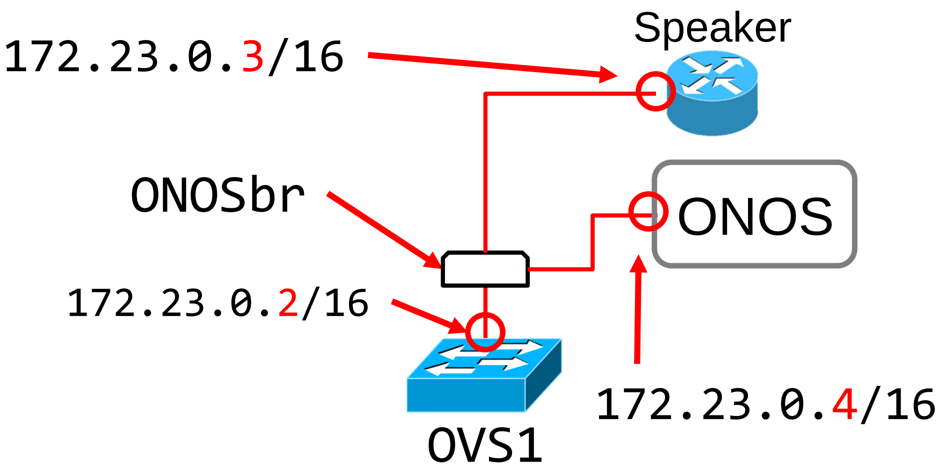
- Setup Container Networks for data plane
- Assume OVS (OVS1), Router (Speaker) and ONOS containers created
- Create docker bridge

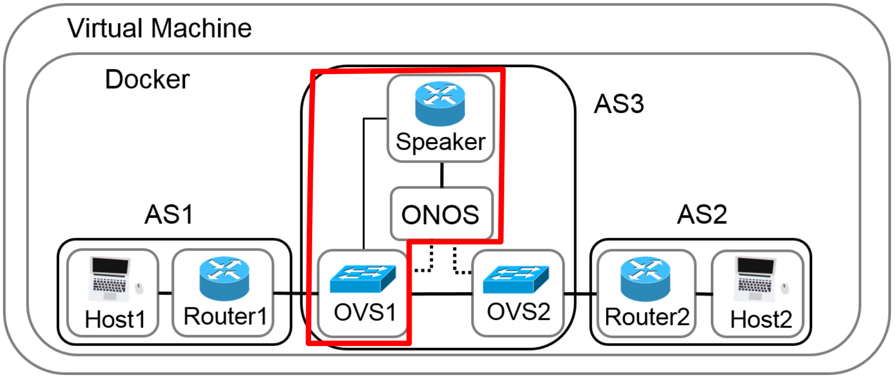
```
~$ sudo docker network create ONOSbr
```

  - ONOSbr: Bridge name

- Connect containers OVS1, Speaker and ONOS to bridge ONOSbr

```
~$ sudo docker network connect ONOSbr ONOS
~$ sudo docker network connect ONOSbr Speaker
~$ sudo docker network connect ONOSbr OVS1
```





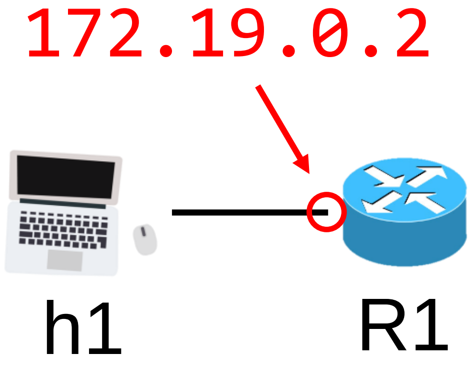
4. Configure Host Gateways

□ Run bash on h1 (h2)

```
~$ sudo docker exec -it h1 bash
```

□ Install Linux routing utilities iproute2 on h1 (h2)

```
/# apt-get update
/# apt-get install -y iproute2
```

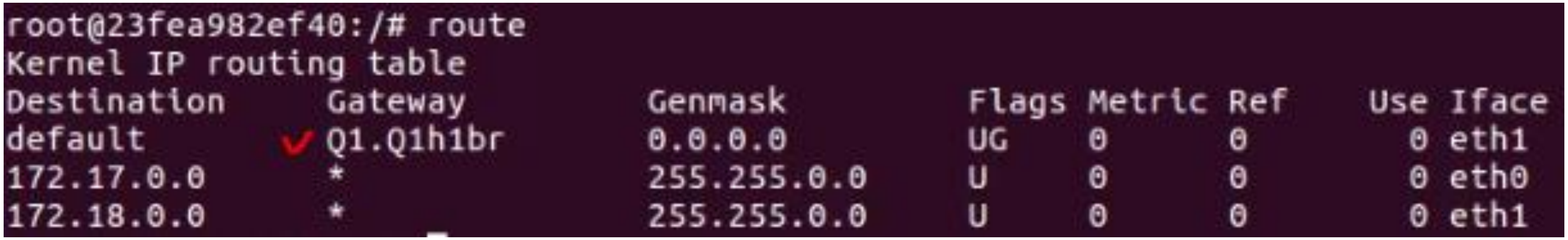


□ Set R1 as default gateway of h1 (h2)

```
/# ip route del default
/# ip route add default via 172.19.0.2
```

□ Check route in h1 (h2)

```
/# route
```



```
root@23fea982ef40:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default       ✓ Q1.Q1h1br       0.0.0.0         UG    0      0        0 eth1
172.17.0.0      *               255.255.0.0     U     0      0        0 eth0
172.18.0.0      *               255.255.0.0     U     0      0        0 eth1
```

5. Setup SDN Network (on same segment)

❑ Run bash on OVS1

```
~$ sudo docker exec -it OVS1 bash
```

■ Set SDN network on same network segment

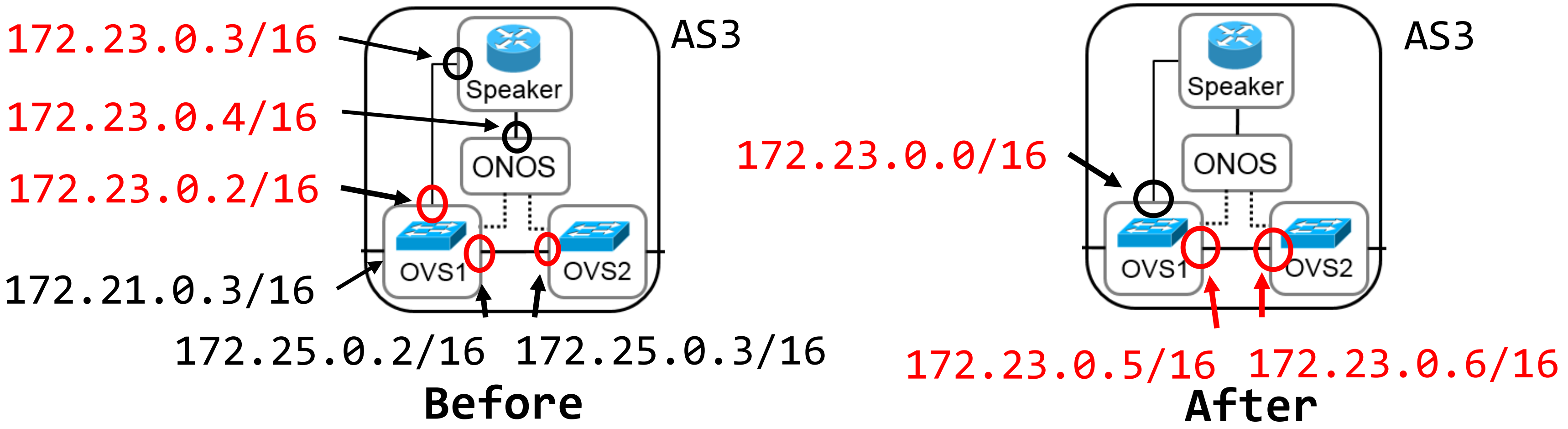
```
/# ifconfig eth2 172.23.0.5 netmask 255.255.0.0
```

❑ Run bash on OVS2

```
~$ sudo docker exec -it OVS2 bash
```

■ Set SDN network on same network segment

```
/# ifconfig eth2 172.23.0.6 netmask 255.255.0.0
```



6. Activate ONOS Applications

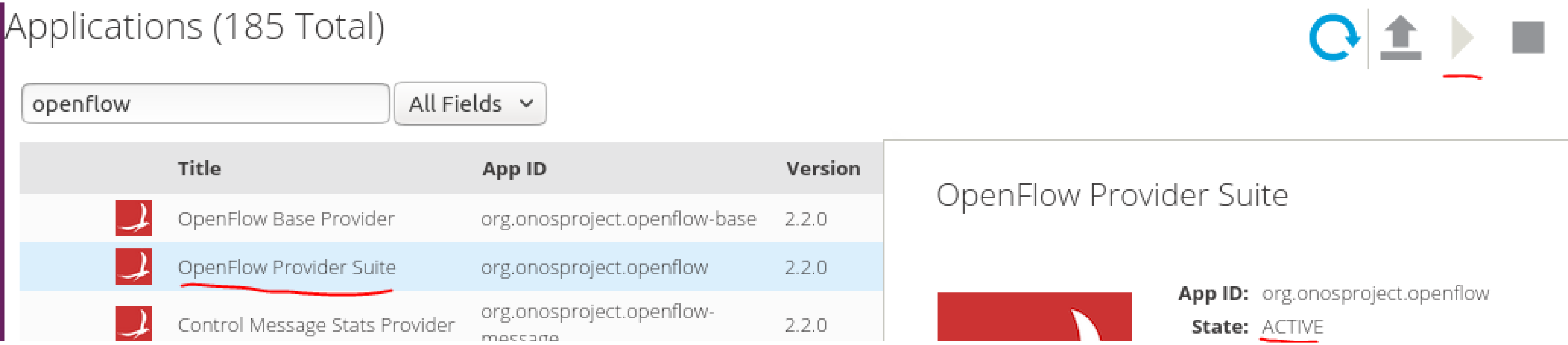
- ❑ Run bash on ONOS

```
~$ sudo docker exec -it ONOS bash
```

- ❑ Use ifconfig to display interfaces in ONOS

```
/# ifconfig
...
eth0 addr:172.17.0.2
```

- ❑ Use ONOS Web GUI to activate application
  - ■ Use browser to enter “http://172.17.0.2:8181/onos/ui”
  - ■ Activate “OpenFlow Provider Suite,” “proxyarp” and “SDN-IP”



Applications (185 Total)

openflow | All Fields

Title	App ID	Version
OpenFlow Base Provider	org.onosproject.openflow-base	2.2.0
OpenFlow Provider Suite	org.onosproject.openflow	2.2.0
Control Message Stats Provider	org.onosproject.openflow-message	2.2.0

OpenFlow Provider Suite

App ID: org.onosproject.openflow

State: ACTIVE

7. Setup OVS – Set Controller

❑ Run bash on OVS1 (OVS2)

```
~$ sudo docker exec -it OVS1 bash
```

❑ Use “ovs-vsctl” to create a OVS bridge

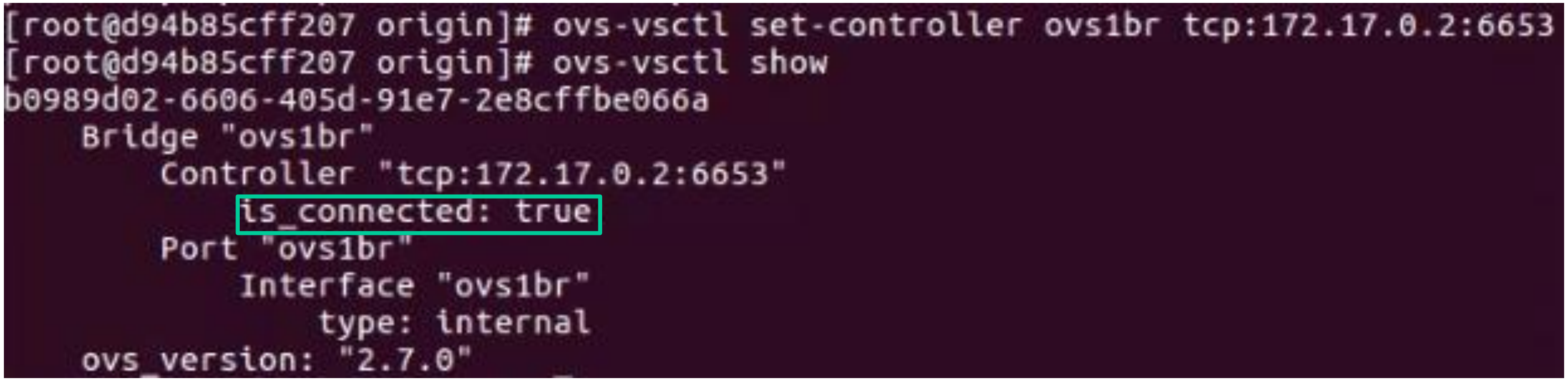
```
/# ovs-vsctl add-br ovs1br
```

❑ Set the controller

```
/# ovs-vsctl set-controller ovs1br tcp:172.17.0.2:6653
```

❑ Print OVS bridge information

```
/# ovs-vsctl show
```

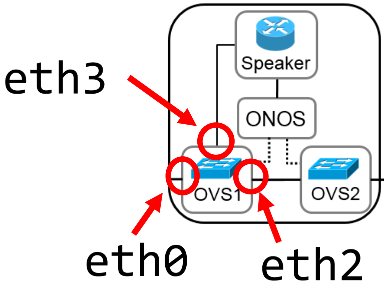


```
[root@d94b85cff207 origin]# ovs-vsctl set-controller ovs1br tcp:172.17.0.2:6653
[root@d94b85cff207 origin]# ovs-vsctl show
b0989d02-6606-405d-91e7-2e8cffbe066a
    Bridge "ovs1br"
        Controller "tcp:172.17.0.2:6653"
            is_connected: true
        Port "ovs1br"
            Interface "ovs1br"
                type: internal
    ovs_version: "2.7.0"
```

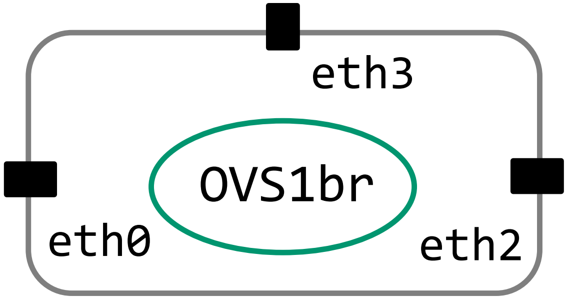
7. Setup OVS – Add OVS1 Port

❑ Add port to OVS1 Bridge

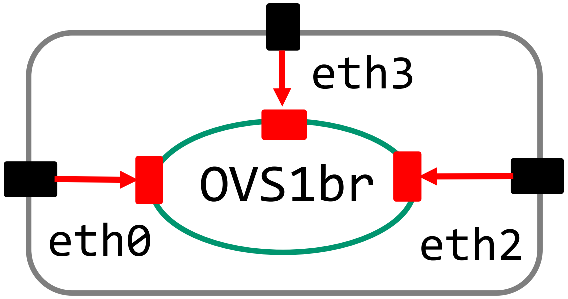
```
/# ovs-vsctl add-port ovs1br eth0
/# ovs-vsctl add-port ovs1br eth2
/# ovs-vsctl add-port ovs1br eth3
```

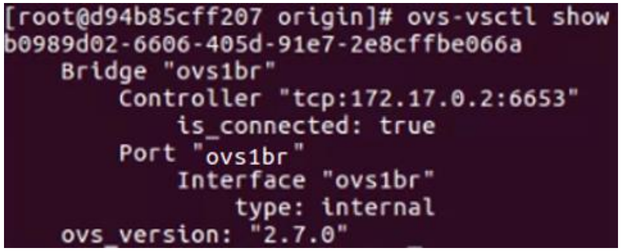


Before

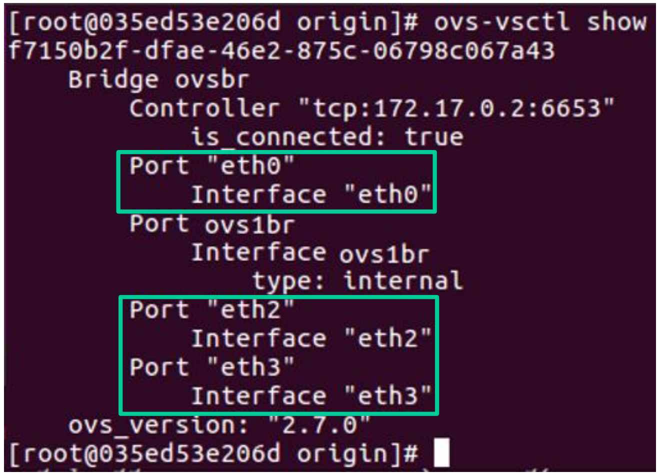


After





```
[root@d94b85cff207 origin]# ovs-vsctl show
b0989d02-6606-405d-91e7-2e8cffbe066a
    Bridge "ovs1br"
        Controller "tcp:172.17.0.2:6653"
            is_connected: true
        Port "ovs1br"
            Interface "ovs1br"
                type: internal
    ovs_version: "2.7.0"
```

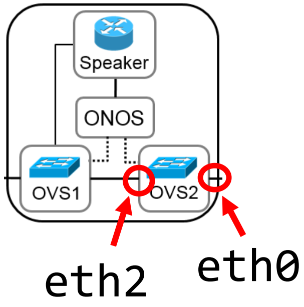


```
[root@035ed53e206d origin]# ovs-vsctl show
f7150b2f-dfae-46e2-875c-06798c067a43
    Bridge ovsbr
        Controller "tcp:172.17.0.2:6653"
            is_connected: true
        Port "eth0"
            Interface "eth0"
        Port ovs1br
            Interface ovs1br
                type: internal
        Port "eth2"
            Interface "eth2"
        Port "eth3"
            Interface "eth3"
    ovs_version: "2.7.0"
[root@035ed53e206d origin]#
```

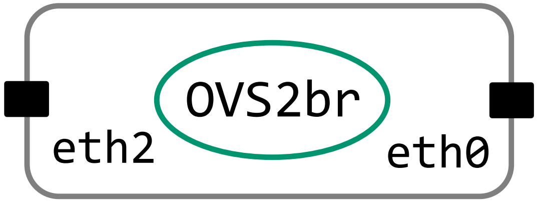
7. Setup OVS – Add OVS2 Port

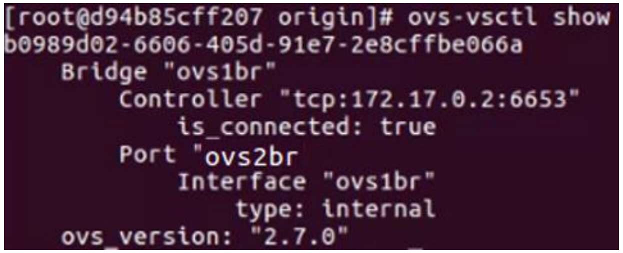
❑ Add port to OVS2 Bridge

```
/# ovs-vsctl add-port ovs2br eth0
/# ovs-vsctl add-port ovs2br eth2
```



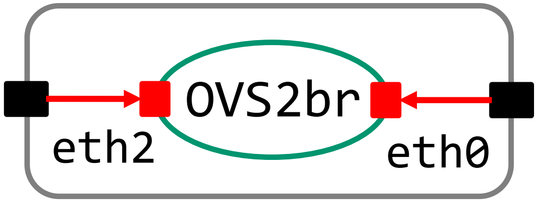
Before

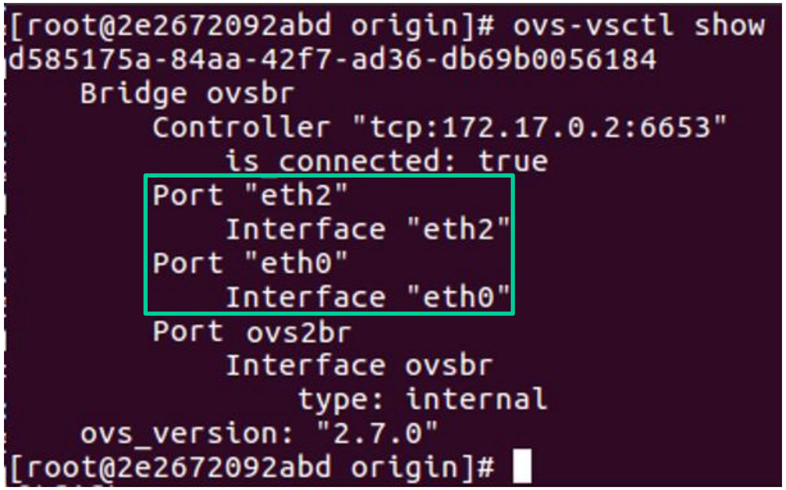




```
[root@d94b85cff207 origin]# ovs-vsctl show
b0989d02-6606-405d-91e7-2e8cffbe066a
    Bridge "ovs1br"
        Controller "tcp:172.17.0.2:6653"
            is_connected: true
        Port "ovs2br
            Interface "ovs1br"
                type: internal
    ovs_version: "2.7.0"
```

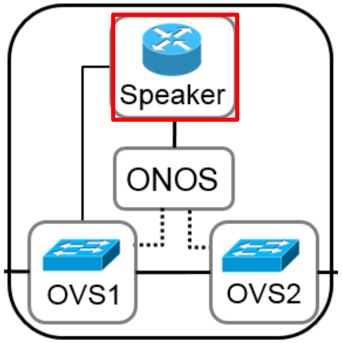
After





```
[root@2e2672092abd origin]# ovs-vsctl show
d585175a-84aa-42f7-ad36-db69b0056184
    Bridge ovsbr
        Controller "tcp:172.17.0.2:6653"
            is_connected: true
        Port "eth2"
            Interface "eth2"
        Port "eth0"
            Interface "eth0"
        Port ovs2br
            Interface ovsbr
                type: internal
    ovs_version: "2.7.0"
[root@2e2672092abd origin]#
```

8. Setup Router - Speaker



- Run bash on Speaker (R1) (R2)

```
~$ sudo docker exec -it Speaker bash
```

- Install vim and quagga on Speaker (R1) (R2)

```
/# apt-get update
/# apt-get install -y vim
/# apt-get install -y quagga
```

- Enable IP forwarding of Speaker (R1) (R2)
  - Edit system control configuration file

    ```
    /# vim /etc/sysctl.conf
    ```

    - Add “net.ipv4.ip_forward = 1” in sysctl.conf
  - Load in sysctl settings from `/etc/sysctl.conf`

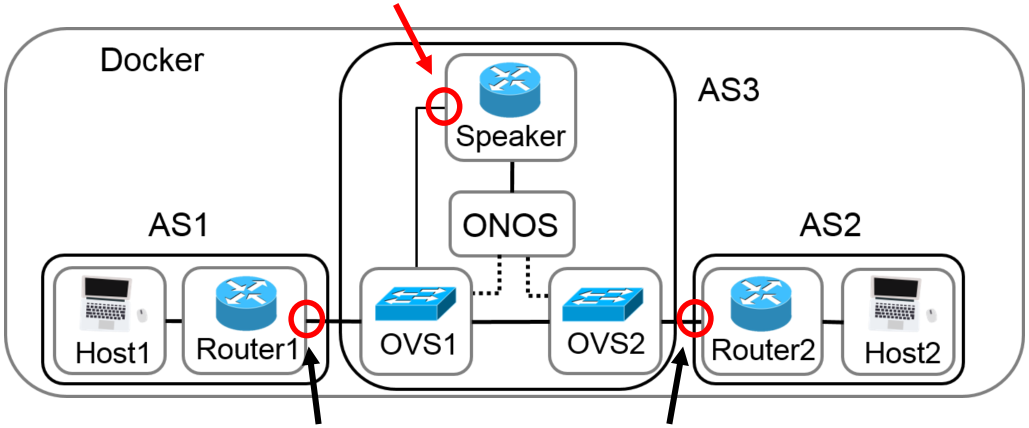
    ```
    /# sysctl -p
    ```

8. Setup Router – Set Speaker IPs for BGP Sessions

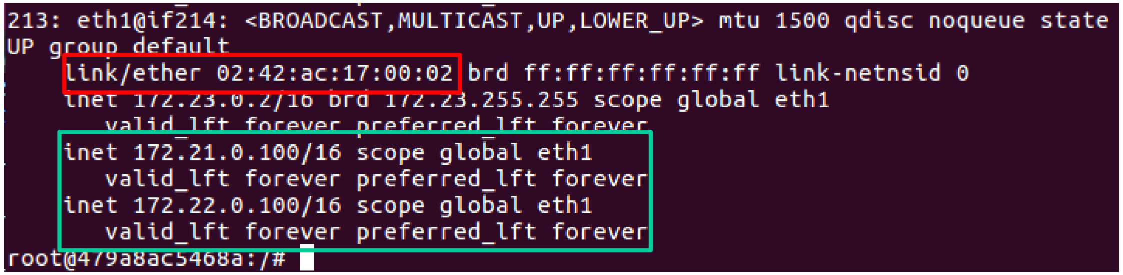
- Create interfaces to receive BGP

```
/# ip addr add 172.21.0.100/16 dev eth1
/# ip addr add 172.22.0.100/16 dev eth1
```

172.21.0.100/16
172.22.0.100/16



172.21.0.2/16 172.22.0.2/16



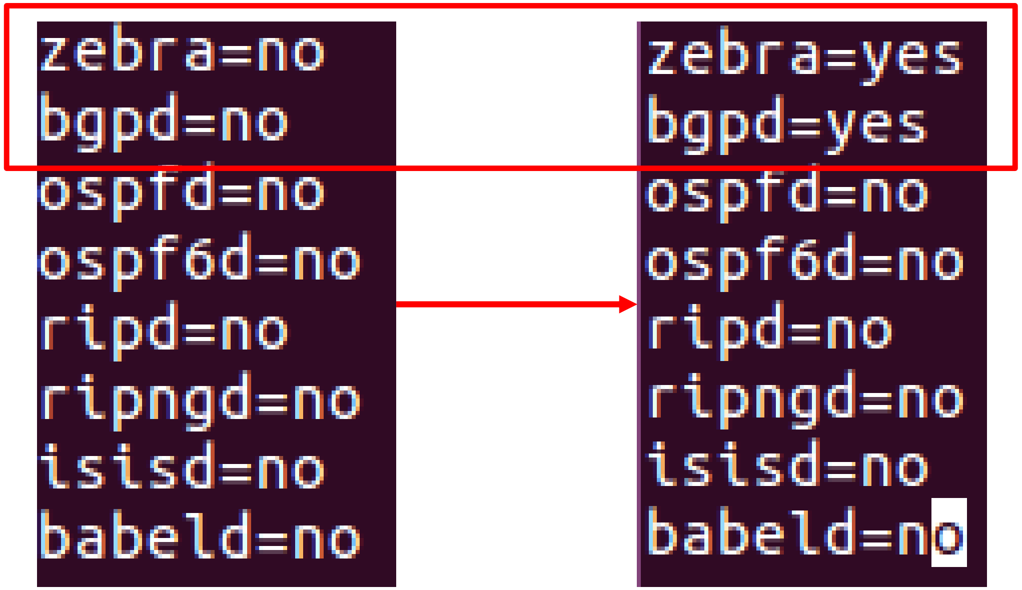
```
213: eth1@if214: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
    link/ether 02:42:ac:17:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.23.0.2/16 brd 172.23.255.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.21.0.100/16 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.22.0.100/16 scope global eth1
       valid_lft forever preferred_lft forever
root@479a8ac5468a:/# 
```

8. Setup Router – Enable Routing Function on Speaker

- Edit Quagga daemons on Speaker (R1) (R2)

```
/# vim /etc/quagga/daemons
```

  - Enable zebra and bgpd daemons
    - ✓ Change zebra and bgpd to Yes



```
zebra=no
bgpd=no
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
```

```
zebra=yes
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
```

8. Setup Router – Set Hostname and Password of zebra

❑ Edit configuration file zebra.conf of Quagga on Speaker (R1) (R2)

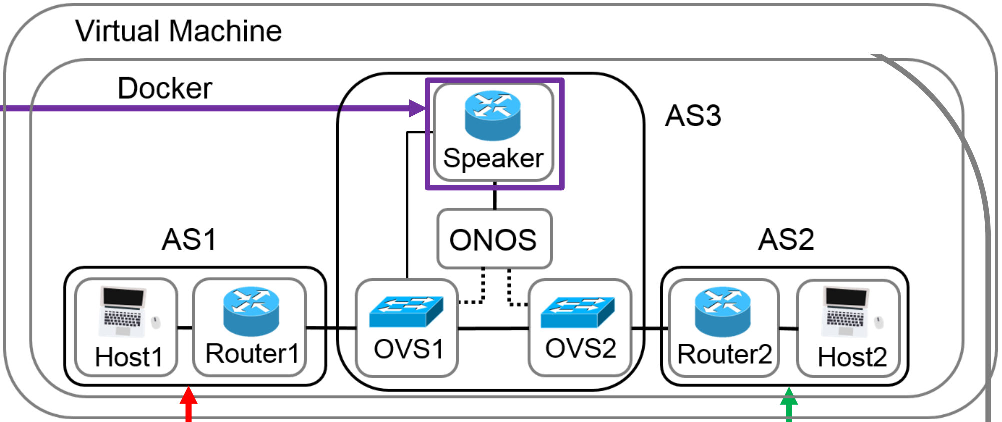
```
/# vim /etc/quagga/zebra.conf
```

❑ Add router name and password in zebra configuration file

```
hostname Speakerzebra (R2zebra) (R1zebra)
password sdnip
log stdout
```

- Hostname: Identifying the zebra on `Speakerzebra` (for shell prompt)
- Password: User access verification

8. Setup Router – Create bgpd.conf on Speaker



```
hostname Speakerbgp
password sdnip
!
router bgp 65000
  bgp router-id 172.17.0.100
  timers bgp 3 9
  !
  neighbor 172.21.0.2 remote-as 65001
  neighbor 172.21.0.2 ebgp-multihop
  neighbor 172.21.0.2 timers connect 5
  neighbor 172.21.0.2 advertisement-interval 5
  !
  neighbor 172.22.0.2 remote-as 65002
  neighbor 172.22.0.2 ebgp-multihop
  neighbor 172.22.0.2 timers connect 5
  neighbor 172.22.0.2 advertisement-interval 5
  !
  ! ONOS
  neighbor 172.17.0.2 remote-as 65000
  neighbor 172.17.0.2 port 2000
  neighbor 172.17.0.2 timers connect 5
```

ASN 65001

ASN 65002

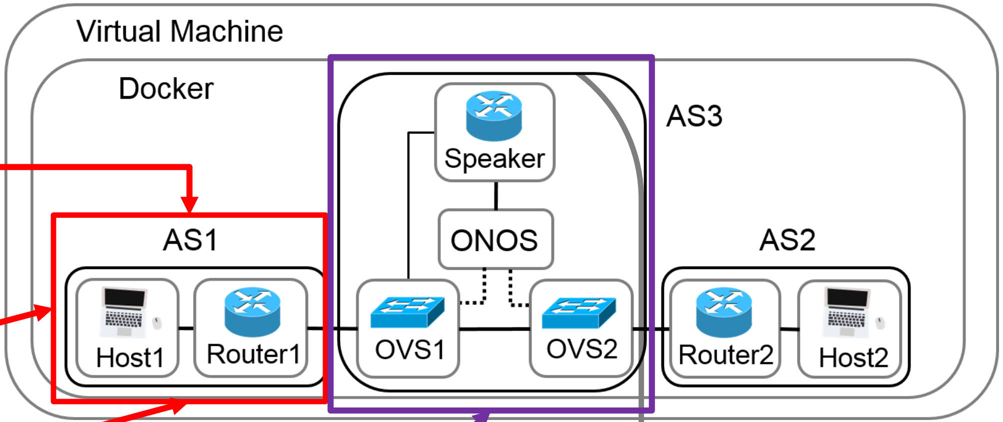
ASN 65000 (SDN-IP)

8. Setup Router – Configure bgpd.conf on R1

- Edit configuration file bgpd.conf of Quagga on R1

```
/# vim /etc/quagga/bgpd.conf
```

```
! BGP configuration for R1
!
hostname R1bgp
password sdnip
!
router bgp 65001
  bgp router-id 172.21.0.2
  timers bgp 3 9
  neighbor 172.21.0.100 remote-as 65000
  neighbor 172.21.0.100 ebgp-multihop
  neighbor 172.21.0.100 timers connect 5
  neighbor 172.21.0.100 advertisement-interval 5
  network 172.19.0.0/16
!
log stdout
```



ASN 65001

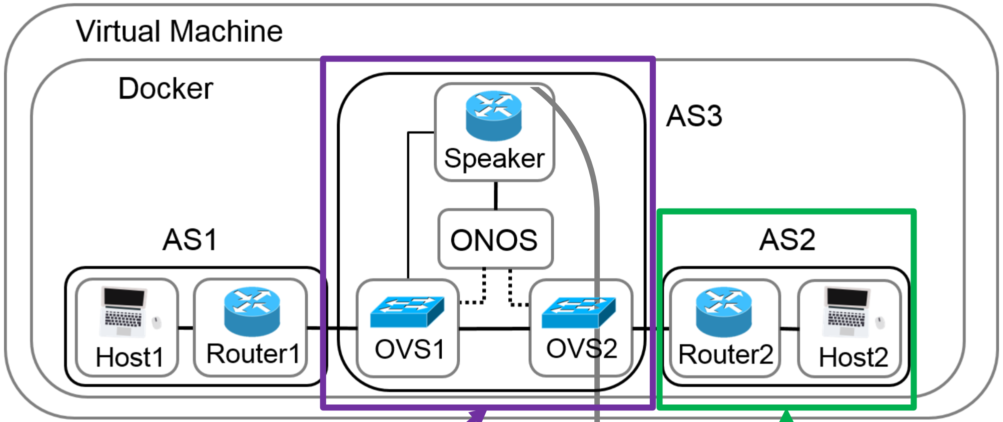
ASN 65000

8. Setup Router – Configure bgpd.conf on R2

- Edit configuration file bgpd.conf of Quagga on R2

```
/# vim /etc/quagga/bgpd.conf
```

```
! BGP configuration for R2
!
hostname R2bgp
password sdnip
!
router bgp 65002
  bgp router-id 172.22.0.2
  timers bgp 3 9
  neighbor 172.22.0.100 remote-as 65000
  neighbor 172.22.0.100 ebgp-multihop
  neighbor 172.22.0.100 timers connect 5
  neighbor 172.22.0.100  advertisement-interval 5
  network 172.20.0.0/16
!
log stdout
```



ASN 65000

8. Setup Router – Restart Quagga on Each Router

- Restart Quagga

```
/# /etc/init.d/quagga restart
```

9. Setup ONOS Network configuration

1) Determine OVS Ports

2) Create Network Configuration (network-cfg.json)

3) Update ONOS Network Configuration

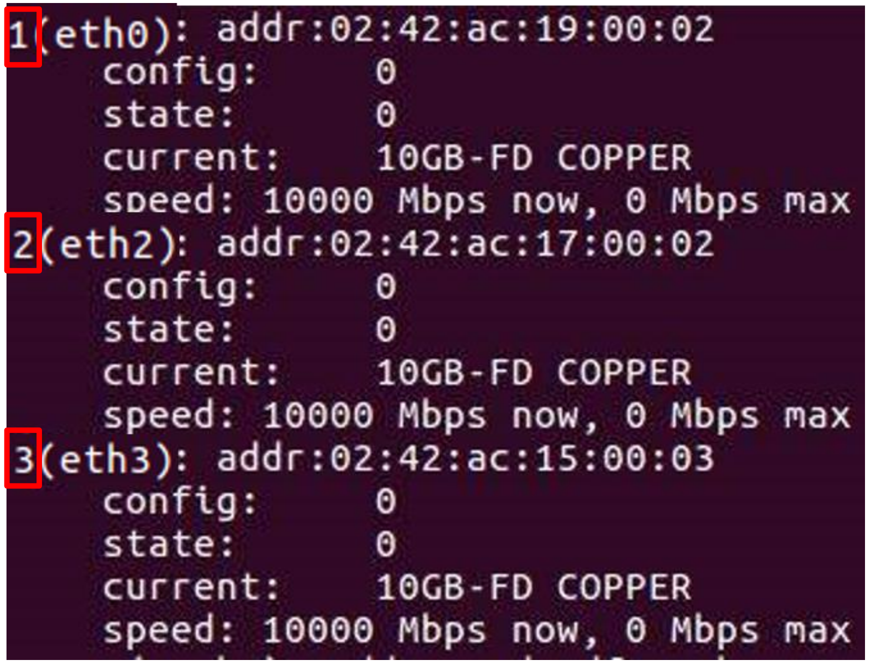
9. Setup ONOS Network Configuration – Determine OVS Ports

- ❑ Run bash on OVS1 (OVS2)

```
~$ sudo docker exec -it OVS1 bash
```

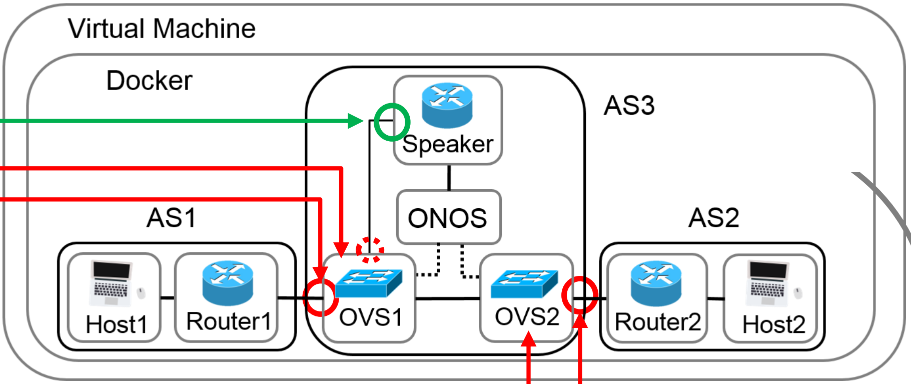
- ❑ Use “ovs-ofctl” to print switch information

```
/# ovs-ofctl show ovs1br
```

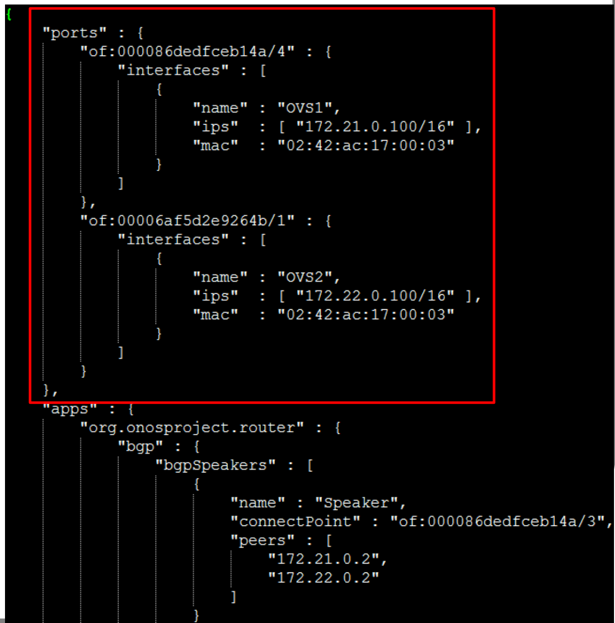


```
1(eth0): addr:02:42:ac:19:00:02
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
2(eth2): addr:02:42:ac:17:00:02
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
3(eth3): addr:02:42:ac:15:00:03
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
```

9. Setup ONOS Network Config.– Create Network Config. (1/2)

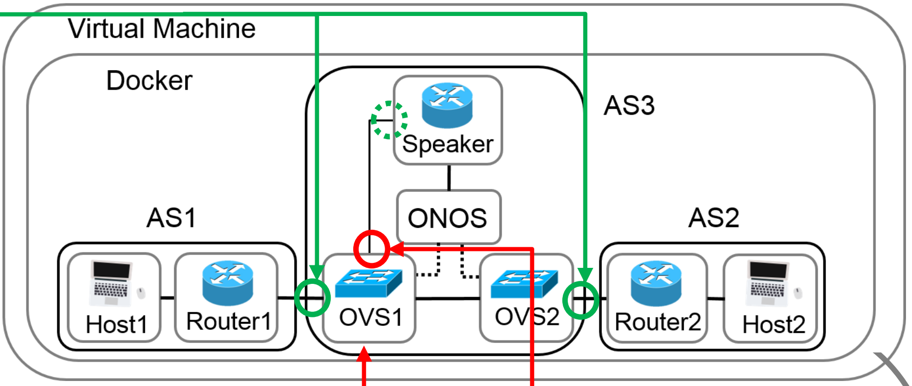


```
{
    "ports" : {
        "of:000086dedfceb14a/1" : {
            "interfaces" : [
                {
                    "name" : "ovs1",
                    "ips"  : [ "172.21.0.100/16" ],
                    "mac"  : "02:42:ac:16:00:06"
                }
            ]
        },
        "of:00006af5d2e9264b/1" : {
            "interfaces" : [
                {
                    "name" : "ovs2",
                    "ips"  : [ "172.22.0.100/16" ],
                    "mac"  : "02:42:ac:16:00:06"
                }
            ]
        }
    },
```

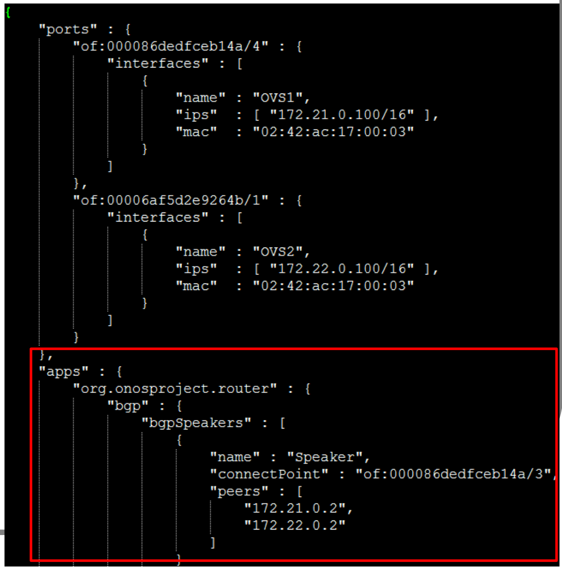


```
{
    "ports" : {
        "of:000086dedfceb14a/4" : {
            "interfaces" : [
                {
                    "name" : "OVS1",
                    "ips"  : [ "172.21.0.100/16" ],
                    "mac"  : "02:42:ac:17:00:03"
                }
            ]
        },
        "of:00006af5d2e9264b/1" : {
            "interfaces" : [
                {
                    "name" : "OVS2",
                    "ips"  : [ "172.22.0.100/16" ],
                    "mac"  : "02:42:ac:17:00:03"
                }
            ]
        }
    },
    "apps" : {
        "org.onosproject.router" : {
            "bgp" : {
                "bgpSpeakers" : [
                    {
                        "name" : "Speaker",
                        "connectPoint" : "of:000086dedfceb14a/3",
                        "peers" : [
                            "172.21.0.2",
                            "172.22.0.2"
                        ]
                    }
```

9. Setup ONOS Network Config. – Create Network Config. (2/2)



```
"apps" : {
       "org.onosproject.router" : {
           "bgp" : {
               "bgpSpeakers" : [
                   {
                       "name" : "speaker",
                       "connectPoint" : "of:000086dedfceb14a/3",
                       "peers" : [
                           "172.21.0.2",
                           "172.22.0.2"
                       ]
                   }
               ]
           }
       }
    }
}
```



```
{
    "ports" : {
        "of:000086dedfceb14a/4" : {
            "interfaces" : [
                {
                    "name" : "OVS1",
                    "ips"  : [ "172.21.0.100/16" ],
                    "mac"  : "02:42:ac:17:00:03"
                }
            ]
        },
        "of:00006af5d2e9264b/1" : {
            "interfaces" : [
                {
                    "name" : "OVS2",
                    "ips"  : [ "172.22.0.100/16" ],
                    "mac"  : "02:42:ac:17:00:03"
                }
            ]
        }
    },
    "apps" : {
        "org.onosproject.router" : {
            "bgp" : {
                "bgpSpeakers" : [
                    {
                        "name" : "Speaker",
                        "connectPoint" : "of:000086dedfceb14a/3",
                        "peers" : [
                            "172.21.0.2",
                            "172.22.0.2"
                        ]
                    }
```

9. Setup ONOS Network Config. – Upload Network Config.

- Update ONOS network configuration

```
~$ curl -u onos:rocks -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' -d
@/home/sdniplab/network-cfg.json
'http://172.17.0.2:8181/onos/v1/network/configuration'
```

- Remove ONOS Network configuration (If necessary)

```
~$ curl -X DELETE --header 'Accept: application/json'
'http://172.17.0.2:8181/onos/v1/network/configuration'
```

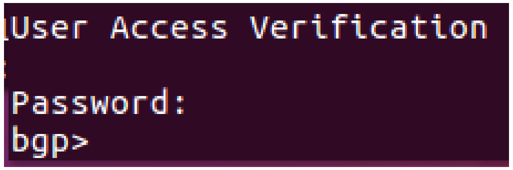
Show bgp Summary in Speaker Container

- ❑ Run bash on Speaker

```
~$ sudo docker exec -it Speaker bash
```

- ❑ Telnet bgpd daemon
  - bgpd listens on port 2605

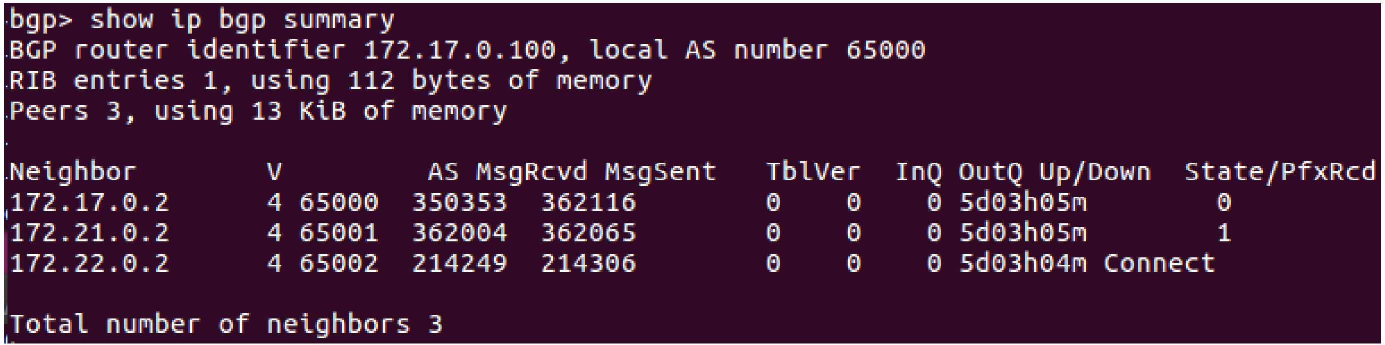
```
~# telnet localhost 2605
```



```
User Access Verification

Password:
bgp>
```

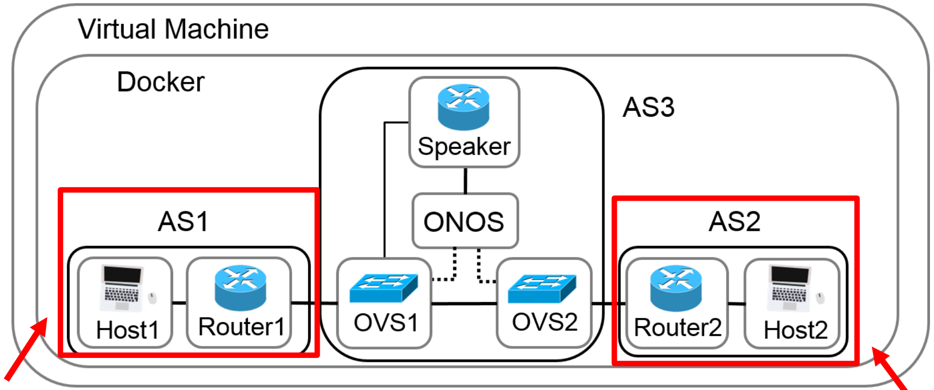
- ❑ Show bgp summary

```
Speakerbgp> show ip bgp summary
```



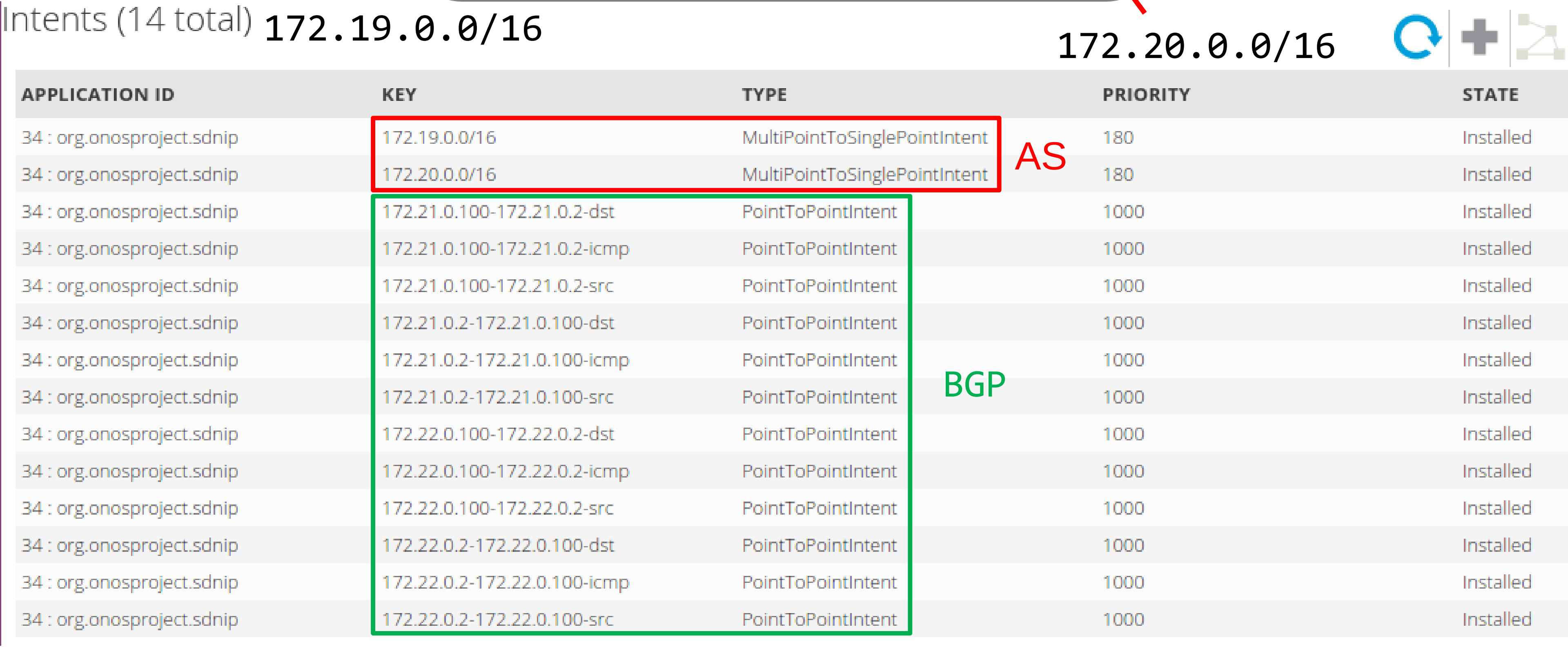
```
bgp> show ip bgp summary
BGP router identifier 172.17.0.100, local AS number 65000
RIB entries 1, using 112 bytes of memory
Peers 3, using 13 KiB of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.17.0.2      4 65000  350353  362116        0    0    0 5d03h05m        0
172.21.0.2      4 65001  362004  362065        0    0    0 5d03h05m        1
172.22.0.2      4 65002  214249  214306        0    0    0 5d03h04m Connect

Total number of neighbors 3
```

Results on ONOS Web GUI



172.19.0.0/16

172.20.0.0/16



Intents (14 total)

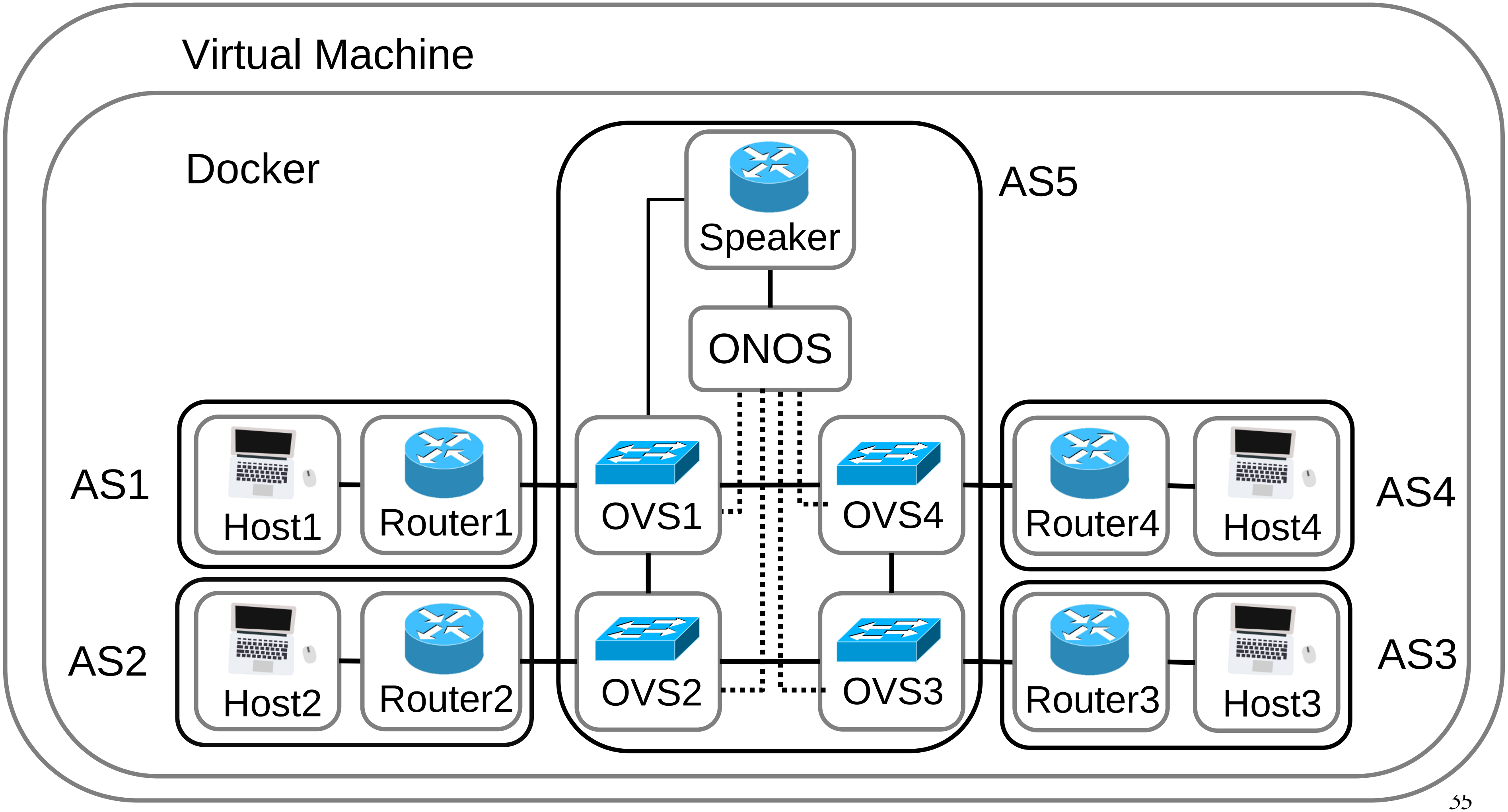
APPLICATION ID	KEY	TYPE	PRIORITY	STATE
34 : org.onosproject.sdnip	172.19.0.0/16	MultiPointToSinglePointIntent	180	Installed
34 : org.onosproject.sdnip	172.20.0.0/16	MultiPointToSinglePointIntent	180	Installed
34 : org.onosproject.sdnip	172.21.0.100-172.21.0.2-dst	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.21.0.100-172.21.0.2-icmp	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.21.0.100-172.21.0.2-src	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.21.0.2-172.21.0.100-dst	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.21.0.2-172.21.0.100-icmp	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.21.0.2-172.21.0.100-src	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.100-172.22.0.2-dst	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.100-172.22.0.2-icmp	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.100-172.22.0.2-src	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.2-172.22.0.100-dst	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.2-172.22.0.100-icmp	PointToPointIntent	1000	Installed
34 : org.onosproject.sdnip	172.22.0.2-172.22.0.100-src	PointToPointIntent	1000	Installed

AS

BGP

Target Topology

- Target Topology of Final Project



Report Submission

- Files
  - A report: FinalProject_<studentID>.pdf
    1. Show topology with IP addresses, interfaces and ASNs
    2. Mark the path eBGP and iBGP in topology
    3. Capture BGP packets (send/receive) in speaker path
    4. Telnet bgpd daemons of Speaker and show summary screenshots
    5. Screenshots topology and intents on the ONOS Web GUI
    6. Write down what you have learned or solved
  - network-cfg.json

- Submission
  - Upload FinalProject _<studentID>.zip to e3
  - Report with incorrect file name or format subjects to not scoring

References

- Open vSwitch Manual
  - http://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt
- SDN-IP Architecture
  - https://wiki.onosproject.org/display/ONOS/SDN-IP+Architecture

Thank You!

謝謝您們的聆聽