SDNFV FINAL PROJECT

SDN Network as Virtual Router

助教：蔣汶儒

sdnta@win.cs.nycu.edu.tw

Deadline: 2024/12/19

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

Review of Labs

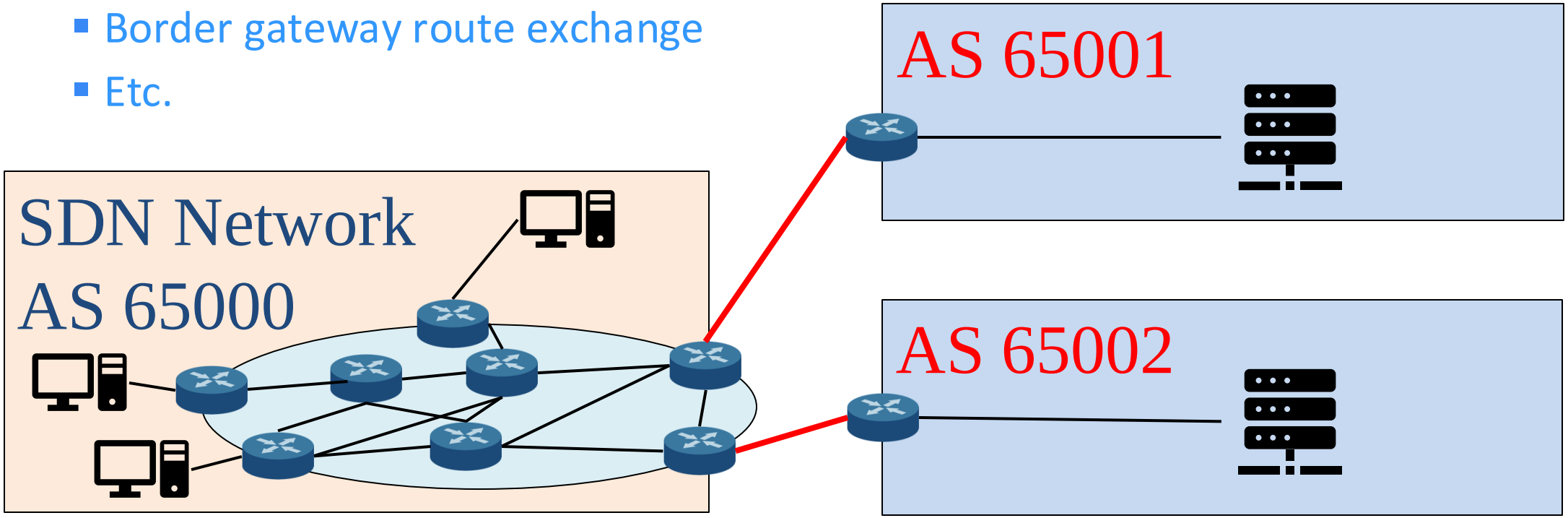
- Lab2
  - ONOS API
  - Flow rules
- Lab3
  - Mac learning
  - Proxy ARP
- Lab4
  - Intent
  - Meter Table
- Lab5 Network Function Virtualization
  - Simulate Autonomous Systems (AS)

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

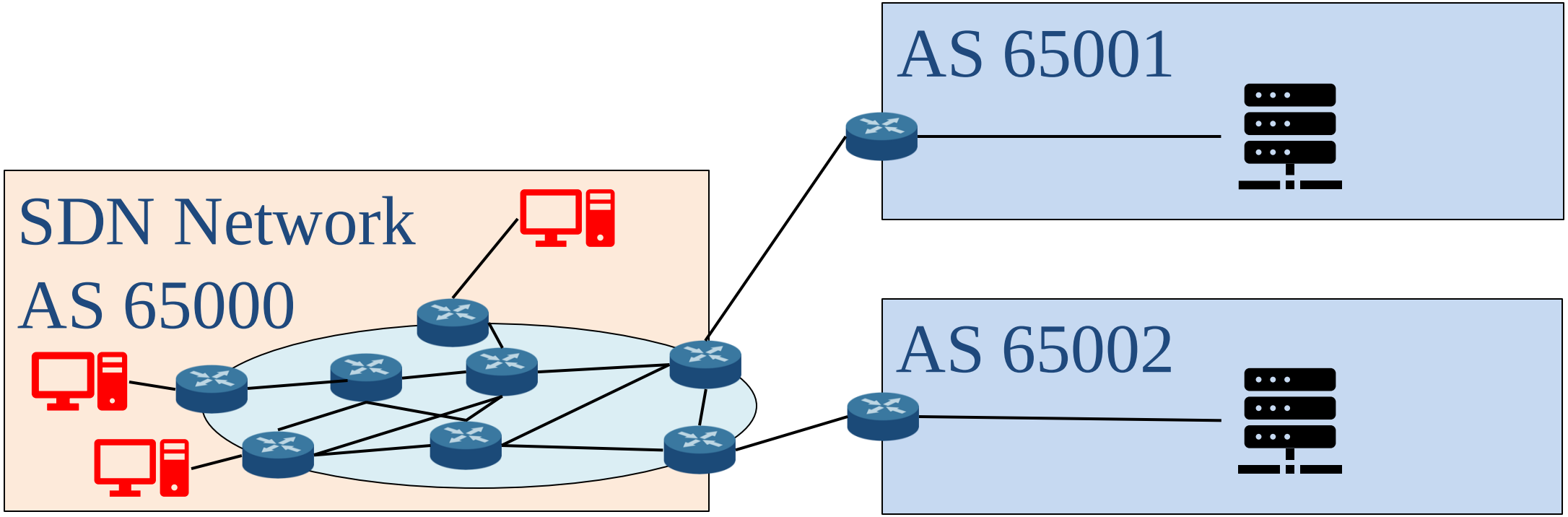
SDN-enabled Virtual Router

- SDN Network with virtual router
  - Use openflow switches and flowrules to simulate router behavior
  - For instance:
    - Layer2 forwarding for next hop communication
    - Border gateway route exchange
    - Etc.



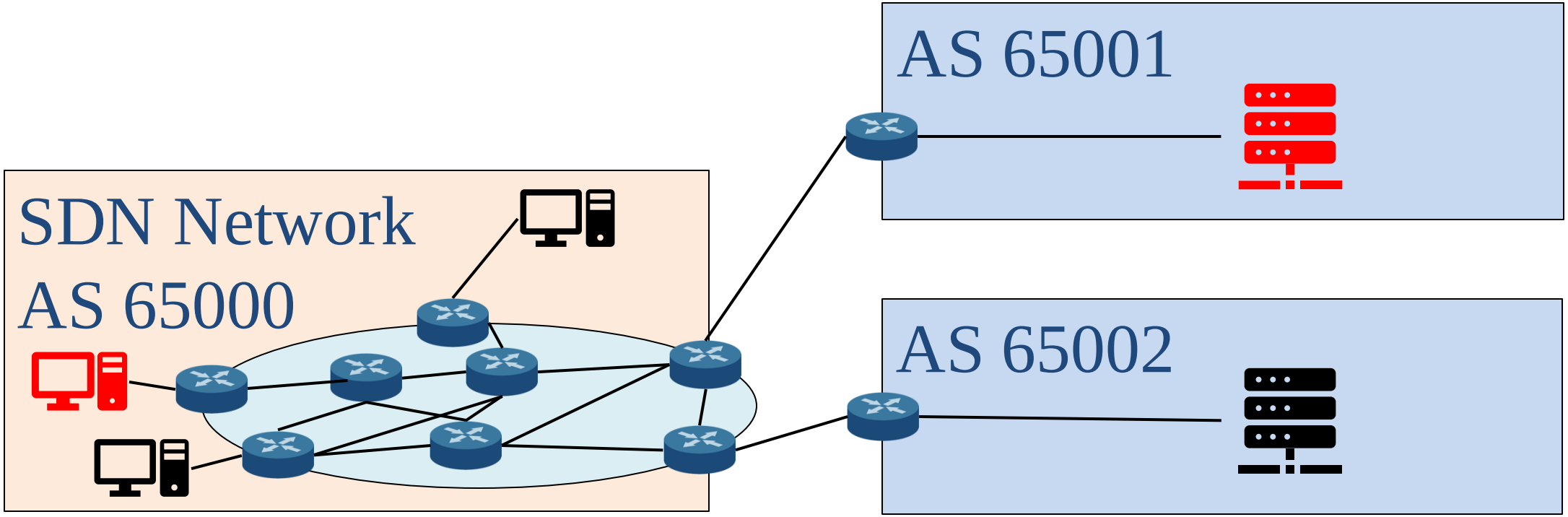
Traffic Types

- Intra-domain Traffics
  - Where **hosts** within the same AS communicates with each other.
  - SDN handles the traffic.



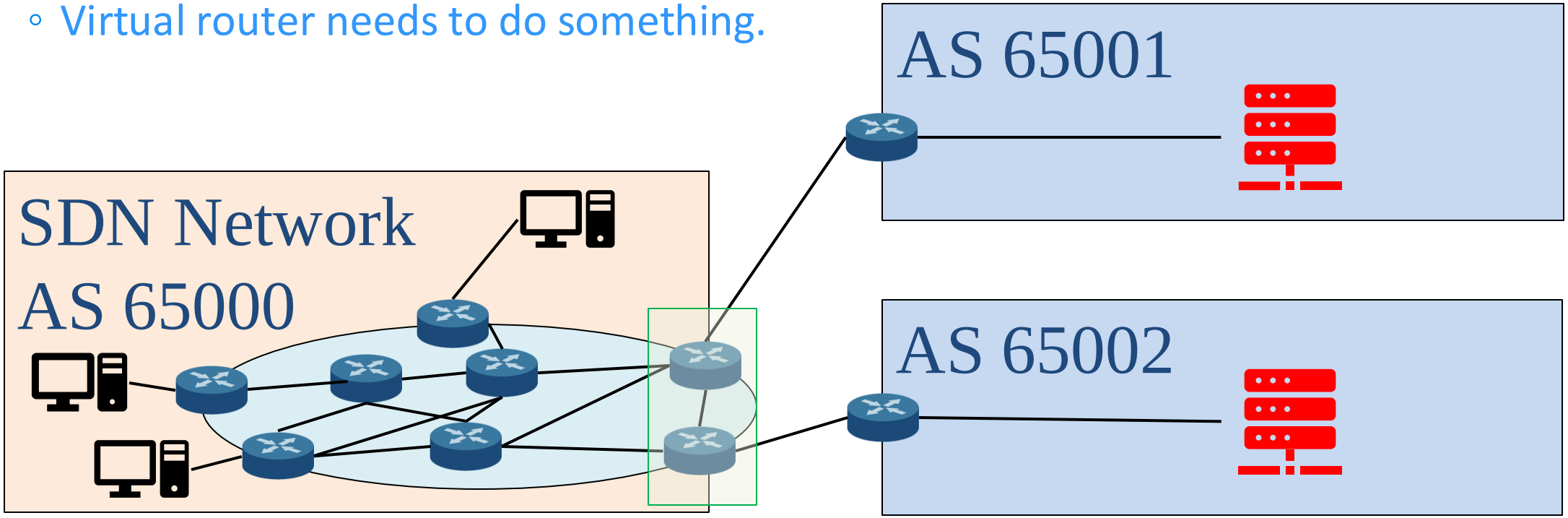
Traffic Types (cont.)

- Inter-domain Traffics
  - Where **an external host** from other domain communicates with **an internal host**.
  - The traffic pass through **gateways**.
  - Virtual router needs to do something.



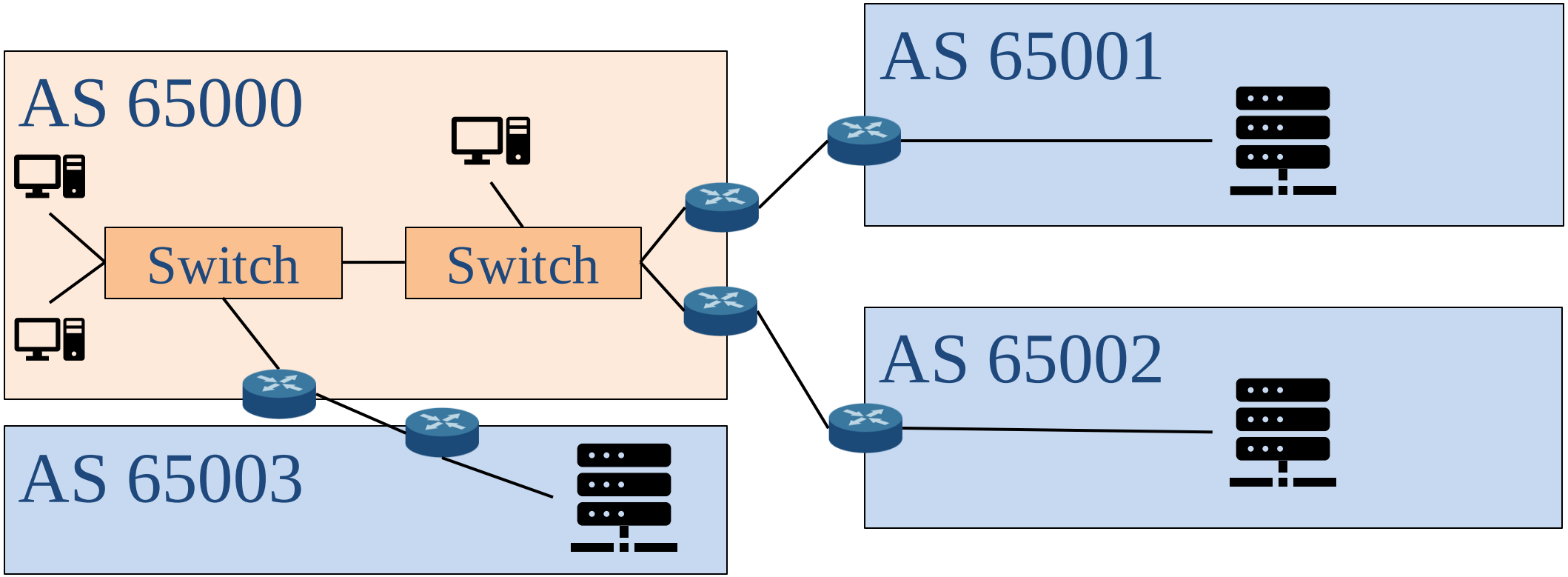
Traffic Types (cont.)

- Transit Traffics
  - Where **hosts** from different domains communicates with one another bypass the SDN network.
  - The traffic pass through virtual router.
  - Virtual router needs to do something.



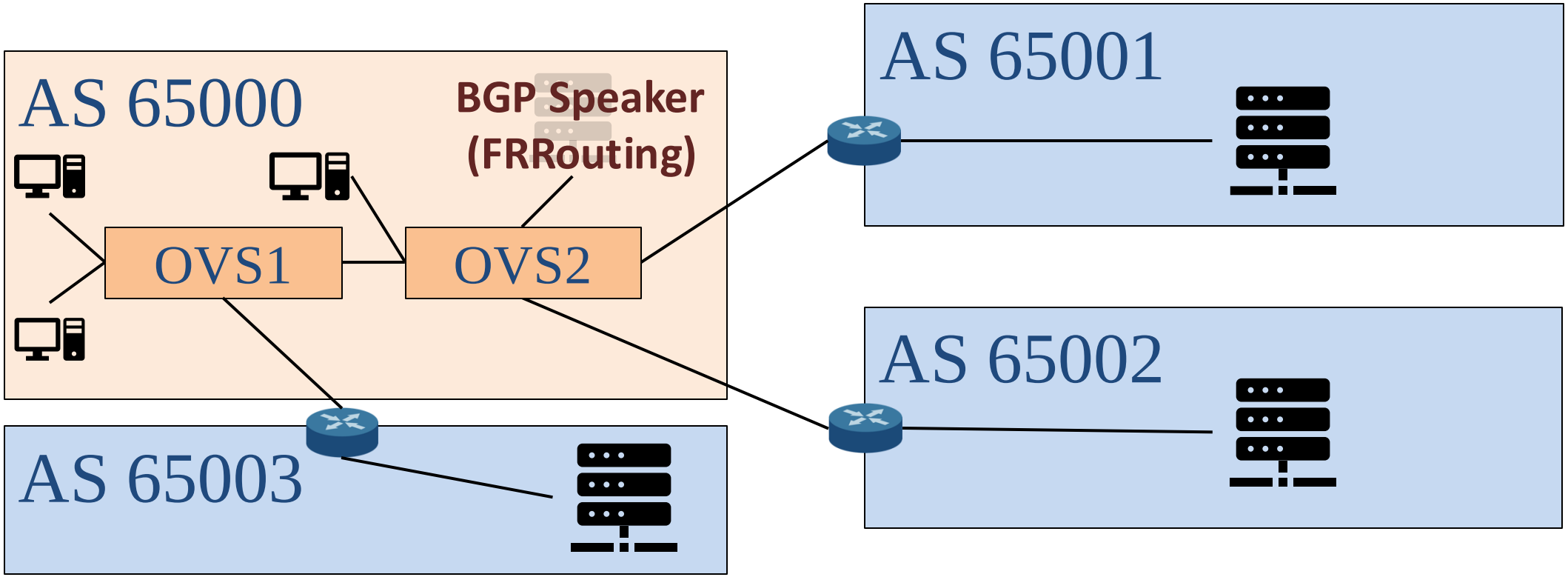
Networks with Physical Routers

- Physical routers
  - 1. Deal with routing decision.
  - 2. Deal with gateway exchange.
- Every edge requires a router, running eBGP and iBGP protocols.



SDN Networks with Virtual Routers

- SDN-enabled Virtual Routers
  - Doesn’t requires router connection to edge.
  - Only one BGP speaker is enough.
  - Doesn’t need a **real gateway.**



OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

Goal

- Intra-domain host communication
  - Handled by Bridge APP
- Inter-domain host communication
  - SDN domain <-> Other domain
- Transit host communication
  - Other domain <-> SDN domain <-> Other domain

vRouter Specification

- Intra AS packet forwarding and packet-in request
  - Lab3
- Arp Reply for devices in AS
  - Lab3
- Inter-domain eBGP traffic topology
  - Lab5
- Routing table maintenance
  - Lab5
- Flowrules for intra/inter/transit domain traffic
  - vRouter APP

➢ IPv4 and IPv6 Dual stack
  - With Additional IPV6 Capability !!!!

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

ARP in IPv6

- ARP are only for IPv4s, how does IPv6 know the MAC address of the target?

- Neighbor Discovery Protocol (NDP)
  - ICMPv6 Messages
  - Neighbor Solicitation (Type 135)
    - Similar to ARP request
  - Neighbor Advertisement (Type 136)
    - Similar to ARP response

Type	Code	Checksum
Content		

ICMPv6 Packet

ARP App Extension for IPv6

- Handle certain packets

```
139             findNDP(pc.inPacket().parsed()).ifPresent(ndPayload -> {
140                 processNDPPacket(pc, ndPayload);
141             });
```

- Example code to determine neighbor solicitation packet type

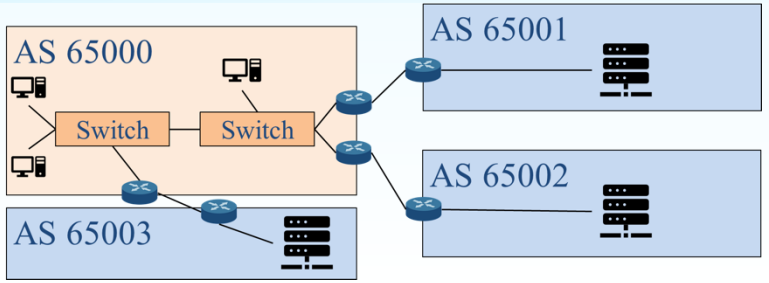
```
302     private Optional<NeighborSolicitation> findNDP(Ethernet packet) {
303         return Stream.of(packet)
304                 .filter(Objects::nonNull)
305                 .map(Ethernet::getPayload)
306                 .filter(p -> p instanceof IPv6)
307                 .filter(Objects::nonNull)
308                 .map(IPacket::getPayload)
309                 .filter(p -> p instanceof NeighborSolicitation)
310                 .map(p -> (NeighborSolicitation) p)
311                 .findFirst();
312     }
```

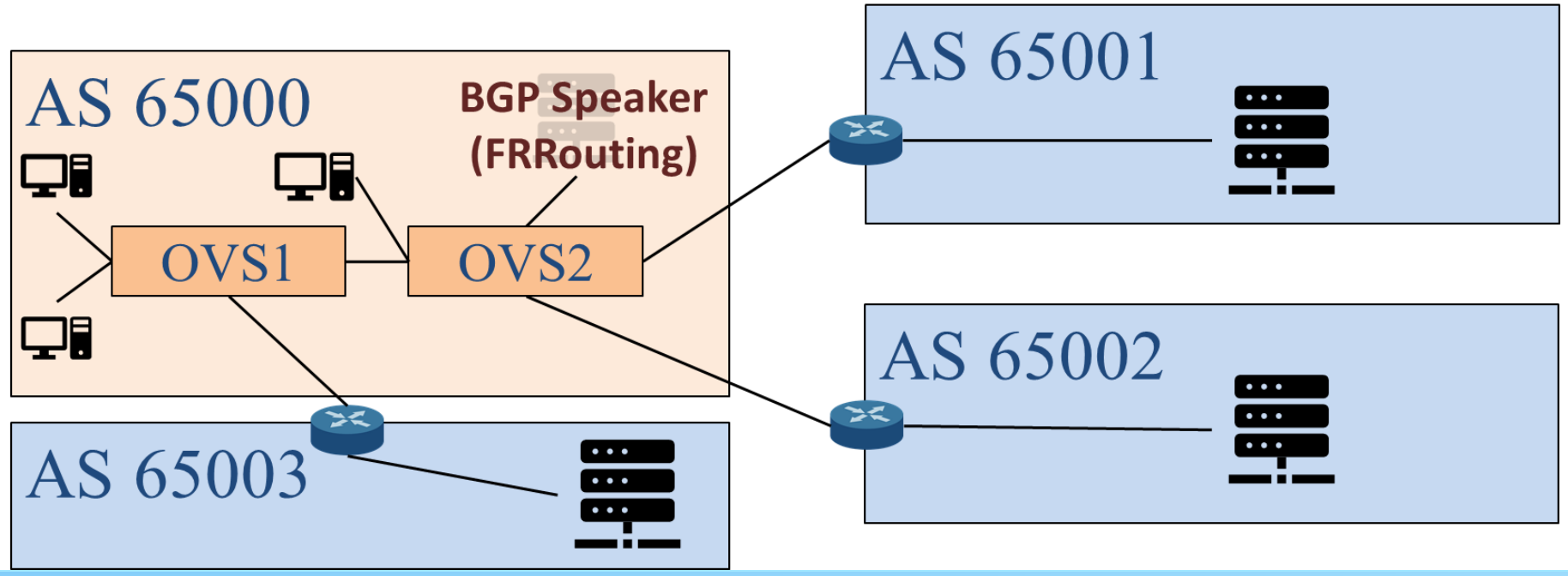
- Simple way to build a neighbor advertisement packet

```
227             outPacket(
228                 pc.inPacket().receivedFrom(),
229                 ByteBuffer.wrap(NeighborAdvertisement.buildNdpAdv(vip6, vmac, packet).serialize()));
```

Virtual Router BGP Connection

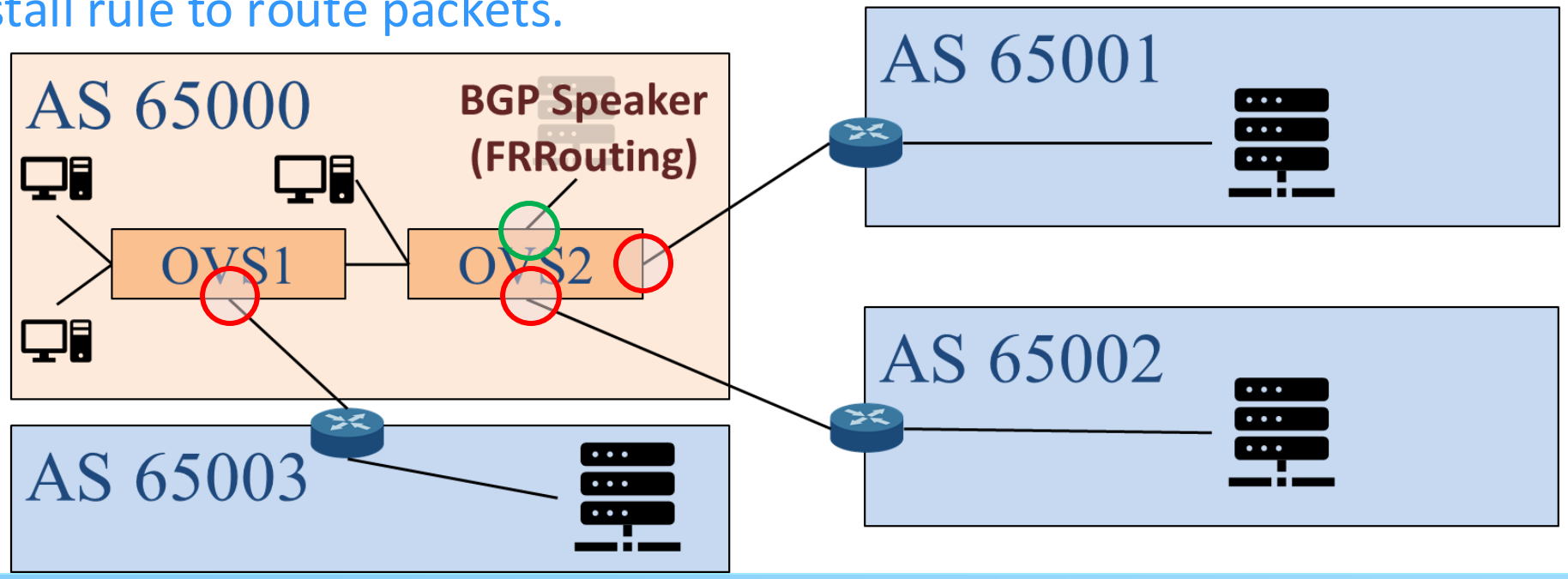
- Physical router:
  - External routers connect with the border gateway.
- Virtual router:
  - External routers connect with BGP Speaker.
  - ➢Need to delegate BGP Speaker IP to edge switch.





BGP Speaker IP Delegation and Routing

1. Delegate BGP speaker IP to the WAN Connect Point on edge switch.
   1) Determine WAN Connect Point.
   2) Config (via netcfg) WAN Connect Point interface.
2. Route packet between BGP speaker Connect Point and WAN Connect Point.
   1. Determine BGP speaker connect point.
   2. Install rule to route packets.



WAN Connect Point Configuration

- Create a configuration file for WAN Connect Point
  - Making external routers think that the BGP Speaker is at the Connect Point.

```
1  {
2      "ports": {
3          "of:0000ceefffee9943/3": {
4              "interfaces": [
5                  {
6                      "name": "intf1",
7                      "ips": [
8                          "192.168.70.1/24",
9                          "fd70::1/64",
10                         "fe80::42:c0ff:fea8:46fd/128"
11                     ]
12                 }
13             ]
14         },
```

Connect Point

Interface Config

***NOTE**

This only gives ONOS controller information of the interface and its IPs. How to make BGP speaker receive packets designated to the IP is your work!

WAN Connect Point Information Retrieval

- Use ONOS Interface Service to retrieve WAN Connect Point.
  - Import Interface service

```
36  import org.onosproject.net.intf.InterfaceService;
```

  - Reference interface service

```
84      @Reference(cardinality = ReferenceCardinality.MANDATORY)
85      protected InterfaceService interfaceService;
```

  - Query WAN Connect Point information from the interface service

```
interfaceService.getMatchingInterface(IpAddress.valueOf("192.168.70.1")).connectPoint()
```

Zebra FIB Pushing

- Zebra supports a Forwarding Information Base (FIB) Push Interface (FPI)
  - FPI allows an external component to learn the forwarding information.
- Forwarding Plane Manager (FPM)
  - Receives FIB
  - Decode FIB into routes
- FIB pushing:
  - FPM establishes a TCP connection with Zebra
  - Zebra pushes FIB to FPM
- In this project, we use ONOS built-in FPM to collect FIB from zebra.

```
karaf@root > app activate org.onosproject.fpm
```

BGP Route Retrieval with Route Service

- Route Service will collect route information via **FPM APP**.
- Routes provided by Route Service contains next hop info for target subnet.

```
karaf@root > routes
                            01:57:40
B: Best route, R: Resolved route

Table: ipv4
B R  Network            Next Hop        Source (Node)
> *  172.17.1.0/24      192.168.63.2    FPM (192.168.70.1)
   Total: 1

Table: ipv6
B R  Network                                  Next Hop                                Source (Node)
> *  2400:6180::/48                           fe80::42:c0ff:fea8:46fd                 FPM (192.168.70.1)
> *  2400:6180:100::/40                       fe80::42:c0ff:fea8:46fd                 FPM (192.168.70.1)
```

- Route Service provide routing table query API.

```
routeService.getRouteTables()
```

ONOS Route Service Usage

- Update dependencies in **pom.xml** file.

```
<dependency>
        <groupId>org.onosproject</groupId>
        <artifactId>onos-apps-route-service-api</artifactId>
        <version>2.7.0</version>
</dependency>
```

- Import methods.

```
import org.onosproject.routeservice.xxx;

@Reference(cardinality = ReferenceCardinality.MANDATORY)
protected RouteService routeService;
```

- Read the docs

https://javadoc.io/doc/org.onosproject/onos-cli/1.8.1/org/onosproject/incubator/net/routing/package-summary.html

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- Zebra and FRRouting Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

Enabling FPM Module

- To enable FPM, you have to set –M fpm in zebra_options at /etc/frr/daemons

```
15  # The watchfrr, zebra and staticd daemons are always started.
16  #
17  bgpd=yes
18  ospfd=no
19  ospf6d=no
20  ripd=no
21  ripngd=no
22  isisd=no
23  pimd=no
24  pim6d=no
25  ldpd=no
26  nhrpd=no
27  eigrpd=no
28  babeld=no
29  sharpd=no
30  pbrd=no
31  bfdd=no
32  fabricd=no
33  vrrpd=no
34  pathd=no
35
36  #
37  # If this option is set the /etc/init.d/frr script automatically loads
38  # the config via "vtysh -b" when the servers are started.
39  # Check /etc/pam.d/frr if you intend to use "vtysh"!
40  #
41  vtysh_enable=yes
42  zebra_options="  -A 127.0.0.1 -s 90000000 -M fpm"
```

FRRouting Configuration

- Configurations in /etc/frr/frr.conf

```
1  ! BGP configuration for frr
2  !
3  frr defaults datacenter
4  !
5  fpm connection ip 192.168.100.1 port 2620
6  !
7  router bgp 65010
8  bgp router-id 192.168.70.1
9  timers bgp 3 9
10 neighbor PEER peer-group
11 neighbor PEER ebgp-multihop
12 neighbor PEER timers connect 5
13 neighbor PEER advertisement-interval 5
14 neighbor 192.168.63.2 remote-as 65011
15 neighbor 192.168.63.2 peer-group PEER
16 neighbor 192.168.70.253 remote-as 65000
17 neighbor 192.168.70.253 password winlab.nycu
18 neighbor 192.168.70.253 peer-group PEER
19 neighbor 192.168.70.253 solo
20 neighbor fd63::2 remote-as 65011
21 neighbor fd63::2 peer-group PEER
22 neighbor fd70::fe remote-as 65000
23 neighbor fd70::fe password winlab.nycu
24 neighbor fd70::fe peer-group PEER
25 neighbor fd70::fe solo
```

FPM connection

Peer Group (template) for neighbors

Use the template

BGP Passwords

Don't advertise the prefix that you received

Announce IPv4 prefix on IPv4 Interface

```
27 address-family ipv4 unicast
28  network 172.16.1.0/24
29  neighbor 192.168.63.2 activate
30  neighbor 192.168.70.253 activate
31  no neighbor fd63::2 activate
32  no neighbor fd70::fe activate
33 exit-address-family
34 !
35 address-family ipv6 unicast
36  network 2a0b:4e07:c4:1::/64
37  neighbor fd70::fe activate
38  neighbor fd63::2 activate
39  no neighbor 192.168.63.2 activate
40  no neighbor 192.168.70.253 activate
41 exit-address-family
42 !
43 log stdout
```

Same as IPv6 Interface

NOTE* Older versions of FRRouting might not work, this is just an example

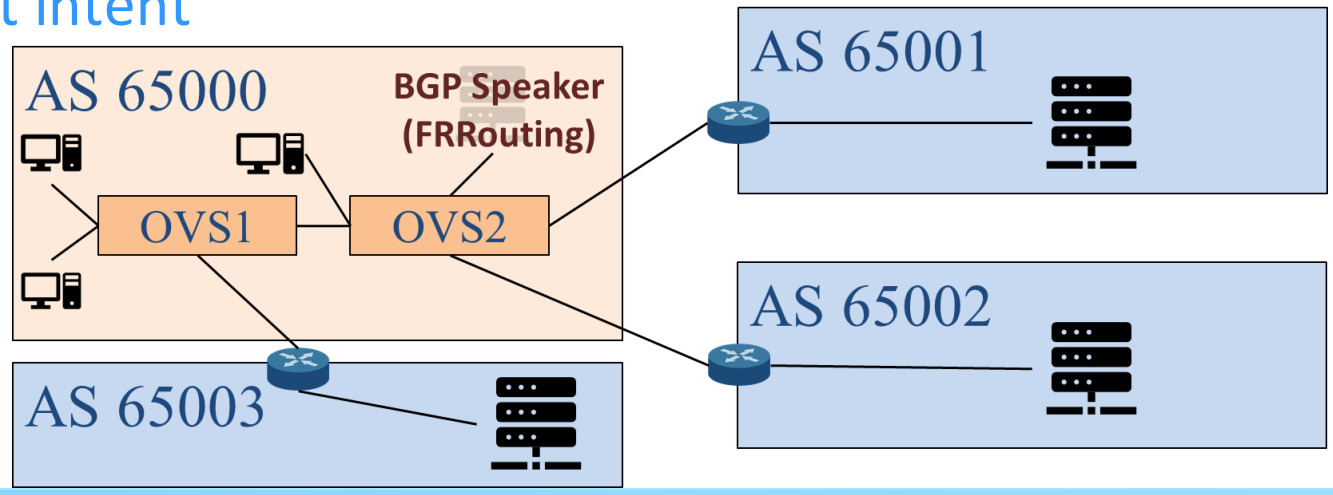
OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

BGP Message Exchange

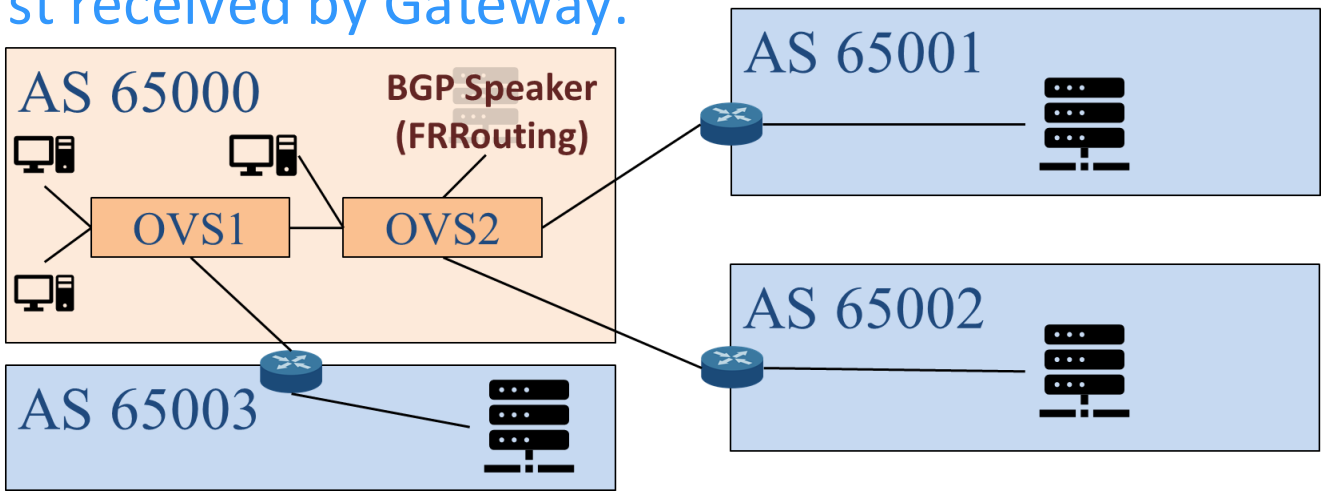
- In order to exchange BGP message with neighbor router
  - Neighbor discovery for L2 connectivity
    - Proxy ARP APP handles ARPs and NDPs on behalf of BGP Speaker.
  - L3 forwarding for BGP Messages

- **L3 forwarding for BGP Messages?**
  - **Incoming**
    - Hint: MultiPointToSinglePoint intent
  - **Outgoing**
    - ????



Virtual Gateway and Inter-domain Routing

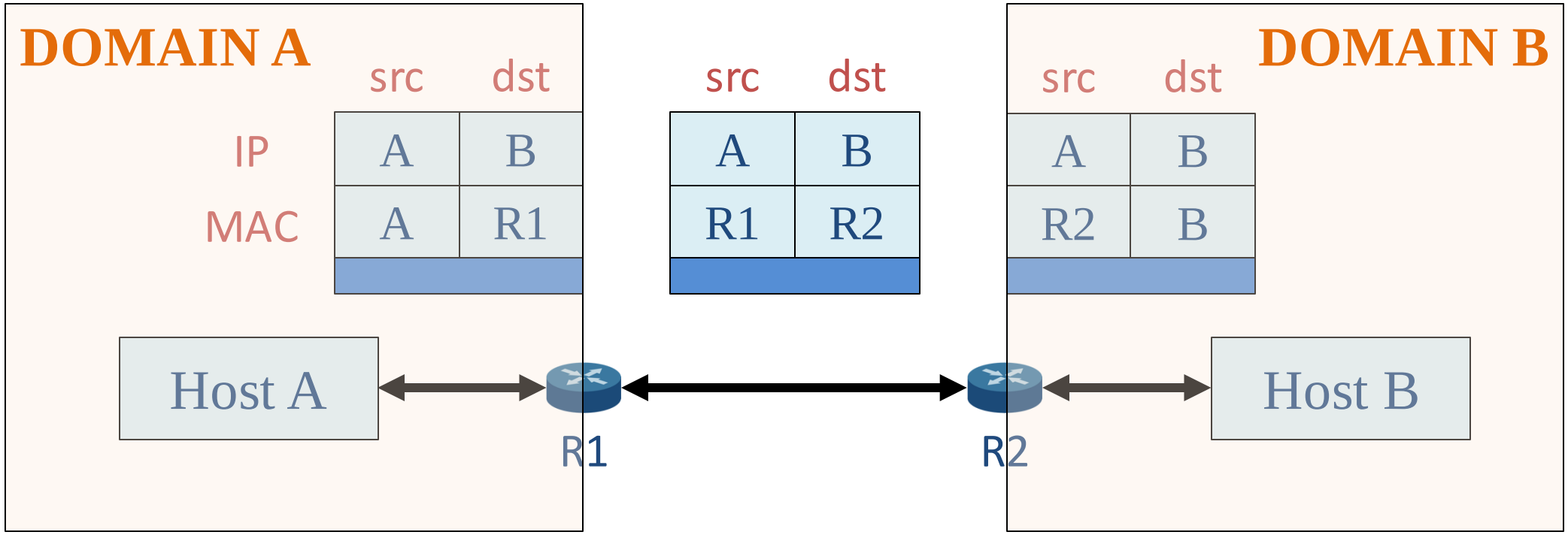
- Gateway and Routing
  - Assume Gateway IP: 192.168.1.254/24
    - Packets originated from 192.168.1.0/24 towards other networks
      - Packet first sent to Gateway.
    - Packet coming from other networks destined 192.168.1.0/24
      - Packet first received by Gateway.



- IP is the logical address of ultimate destination.
  - But, MAC is the physical address of the next hop.

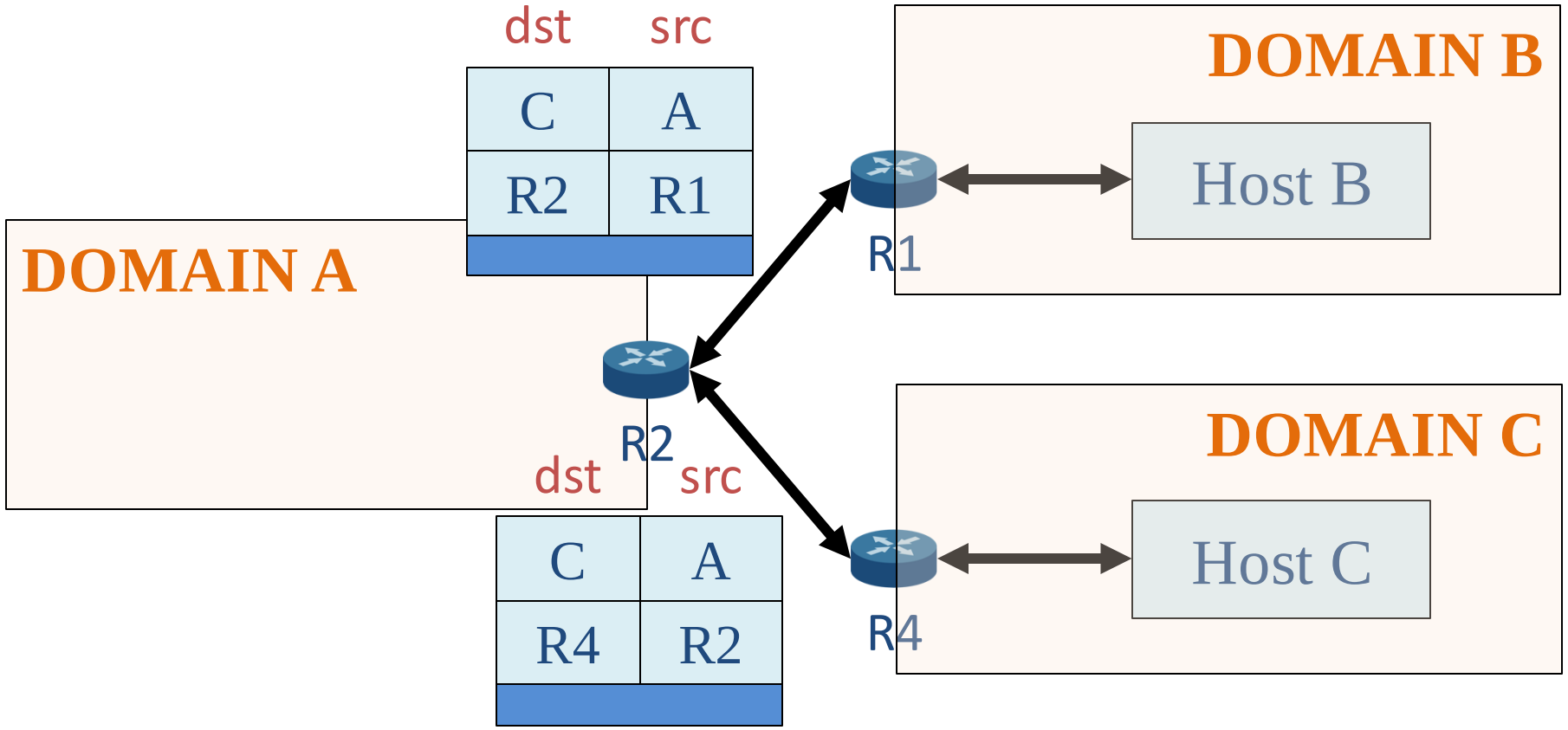
Gateway Traffic Handling Example

- Any packets within the domain only knows about the gateway's MAC.
- After analyzing the information (IP), it will change the according MAC and sends the packet out.



Transit Traffics

- Transit traffics are in fact two interdomain traffics.



OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Project Information and Installation
- Supplement
- Scoring Criteria
- Reference

Linux Network Brief Introduction

- **Network Namespaces**
  - Provide a way to create isolated network environments within a Linux system.
  - Allow processes to have their own network stack, including interfaces, routing tables, and firewall rules.

- **Each Container have it's own Network Namespace.**

- A network **Bridge** is a kernel created logical L2 switch

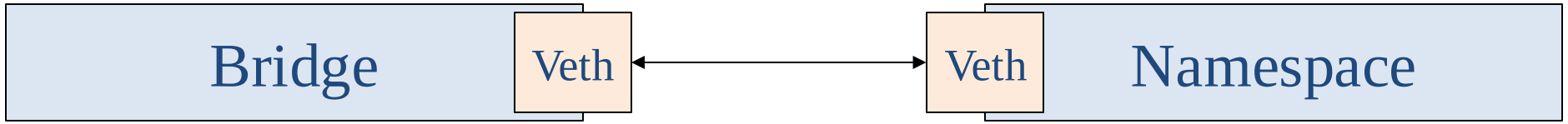
- **Veth** devices, short for virtual Ethernet devices
- Use **veth pairs** to connect Network Namespaces or Bridges together.

Linux Network Brief Introduction (cont.)

- Mapping to physical instruments.

- Namespace = node (computer/server)
- Veth pair = 2 network interface cards (NIC) that connects to each other
- Bridge = switch

- If you want to connect your computer to a switch
  - Create a veth pair (Create 2 NIC that connects to each other)
  - Connect one NIC to your namespace
  - Connect the other one to your bridge



Linux Network Brief Introduction (cont.)

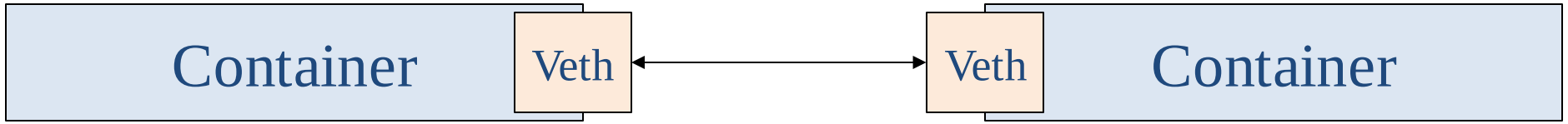
- Mapping to physical instruments.

- Namespace = node (computer/server)
- Veth pair = 2 network interface cards (NIC) that connects to each other
- Bridge = switch
- How to find a container's namespace (ns)?
  - Locate docker pid

```
n0ball@SDN-NFV:~/workspace$ docker inspect -f '{{.State.Pid}}' $(docker ps -aqf "name=sdnfv-demo")
1761815
```

  - It is at file `/proc/$pid/ns/net`

- Similarly you can connect two namespaces (containers) with the same mechanism.



Ubuntu IP Command Introduction

- Normally, we can use `ip netns exec` command to execute commands inside a namespace; however, it will only search ns for directories in `/var/run/netns`

- Two ways to run `ip netns exec` in container namespace
  - Create a soft link `ln -sfT /proc/$pid/ns/net /var/run/netns/$pid`
  - Use nsenter command `nsenter -t $pid -n <command>`

- Useful ip commands
  - `ip link add <name> type <type>`: Create a NIC by the type.
  - `ip link set <name> up`: Bring up (enable) the NIC.
  - `ip address add <ip> dev <name>`: Add an ip address to the NIC.
  - `ip route show`: Show current routes.
  - `ip route add {<ip> | default} via {ip}: Add a route.

Docker Network Namespace Introduction

```
n0ball@SDN-NFV:~/workspace$ docker run -d --rm --name sdnfv-demo alpine:3.2 sleep 10m
46cc48421aa17e80733c73cc93ff6cc3567a25edcf123336111f930553b7c27a
n0ball@SDN-NFV:~/workspace$ docker inspect -f '{{.State.Pid}}' $(docker ps -aqf "name=sdnfv-demo")
1768219
n0ball@SDN-NFV:~/workspace$ sudo ln -s /proc/1768219/ns/net /var/run/netns/1768219
```

Create a container named sdnfv-demo

Find the pid of the container

Make soft link so that ip netns can find container ns

```
n0ball@SDN-NFV:~/workspace$ sudo ip netns exec 1768219 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1520: eth0@if1521: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Show interface information of ns

```
n0ball@SDN-NFV:~/workspace$ sudo ip netns exec 1768219 ip link add eth-test type dummy
```

Create a dummy NIC using netns command

```
n0ball@SDN-NFV:~/workspace$ docker exec sdnfv-demo ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth-test: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 1a:e4:0a:9a:56:c7 brd ff:ff:ff:ff:ff:ff
1520: eth0@if1521: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

NIC is created inside the container

Show interface information of the container

Tunnel and VXLAN

- A Tunnel send data over a network by encapsulating packets within other packets.
  - Commonly used to connect different networks, provide secure communication, bypass firewalls.
  - Applications
    - Virtual Private Networks (VPN)
    - Generic Routing Encapsulation (GRE)
    - Virtual Extensible LAN (VXLAN)
  - Types
    - Layer 2: Transmit data that is higher or equal to Layer 2
    - Layer 3: Transmit data that is higher or equal to Layer 3

- VXLAN (L2 Tunnel)
  - Designed to help build large, scalable L2 networks over existing L3 networks

Wireguard

- TA will provide a Wireguard configuration file
  - **Remember, the last number of your IP is your ID (x)**
- Run the follow command to install wireguard

```
n0ball@SDN-NFV:~/workspace$ apt install -y wireguard
```

- Copy the configuration file to wg0

```
n0ball@SDN-NFV:~/workspace$ cp xxx.conf /etc/wireguard/wg0.conf
```

- Bring up the Wireguard interface

```
n0ball@SDN-NFV:~/workspace$ sudo wg-quick up wg0
```

- Check if Wireguard is good
  - Wireguard gateway

This is Wireguard gateway

```
n0ball@SDN-NFV:~/workspace$ ping 192.168.61.254
PING 192.168.61.254 (192.168.61.254) 56(84) bytes of data.
64 bytes from 192.168.61.254: icmp_seq=1 ttl=64 time=10.2 ms
```

  - VXLAN Target

```
n0ball@SDN-NFV:~/workspace$ ping 192.168.60.200
PING 192.168.60.200 (192.168.60.200) 56(84) bytes of data.
64 bytes from 192.168.60.200: icmp_seq=1 ttl=63 time=12.6 ms
```

This is your x

```
1  # AUTOGENERATED FILE - DO NOT EDIT
2  # This file uses wg-quick format.
3  # See https://man7.org/linux/man-pages/man8/wg-quick.8.html#CONFIGURATION
4  # Lines starting with the -WGP- tag are used by
5  # the WireGuard Portal configuration parser.
6
7  # -WGP- WIREGUARD PORTAL CONFIGURATION FILE
8  # -WGP- version unknown
9
10 [Interface]
11 # -WGP- Peer: EgPpCagJ1r6mBzTYjYrnUQt0bC6Xc41a8Ga31gcdbmI=
12 # -WGP- Created: 2024-10-27 15:00:59.085921029 +0000 UTC
13 # -WGP- Updated: 2024-10-27 15:00:59.090268647 +0000 UTC
14 # -WGP- Display name: Peer EgPpCagJ stu
15 # -WGP- PublicKey: EgPpCagJ1r6mBzTYjYrnUQt0bC6Xc41a8Ga31gcdbmI=
16 # -WGP- Peer type: client
17
18 # Core settings
19 PrivateKey = wAr/OZnGxxxxxxxxxxxxxxxxxxxxYGlkTx3xBxxxx
20 Address = 192.168.61.1/32
21
22 # Misc. settings (optional)
23 MTU = 1420
24
25 # Interface hooks (optional)
26
27 [Peer]
28 PublicKey = yhjlxxxxkMOuD5xxxxxEbxxxxHsOxxxxxaUa+6y5n1xxxx=
29 Endpoint = 10.10.100.250:51820
30 AllowedIPs = 192.168.60.0/23,fe60::/64
31 PresharedKey = tweIi0pRxxxxQyxxxxoY7E1xxxBXDMxxxt6L+5xxxx=
32 PersistentKeepalive = 16
```

This is your ID (x)

Some wireguard version requires port

Maximum Transmission Unit (MTU)

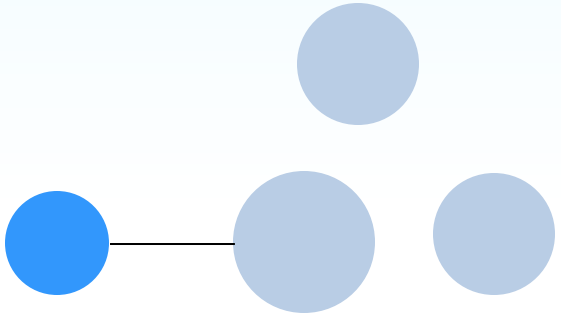
- The largest size (in bytes) of a network packet that can be transmitted over a particular interface or network medium without fragmentation.
- Default to 1500 Bytes.

- What Happens if Packet Size Exceeds the MTU?
  - Fragmentation
  - Drop (Especially IPv6)

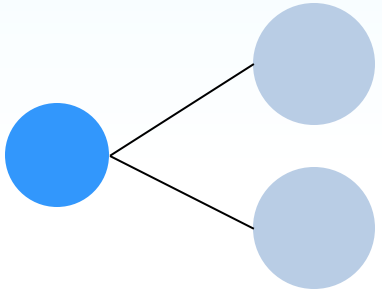
- Suggested subtraction of MTU due to encapsulation
  - VXLAN: 50 Bytes
  - Wireguard: 80 Bytes

Autonomous System (AS)

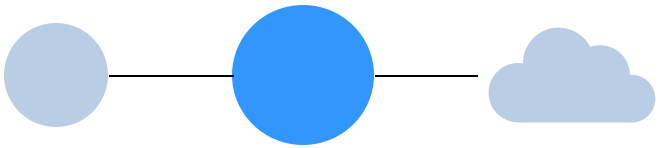
- AS Types



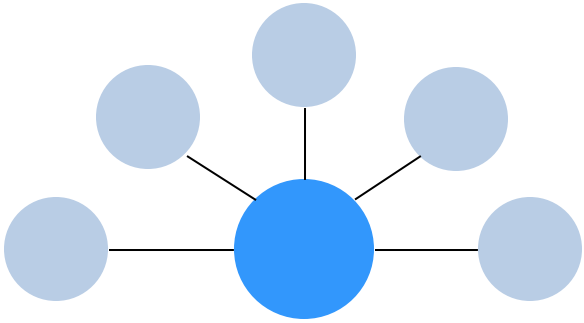
Stub



Multi Homed



Transit



Internet Exchange Point

ISP Characteristic

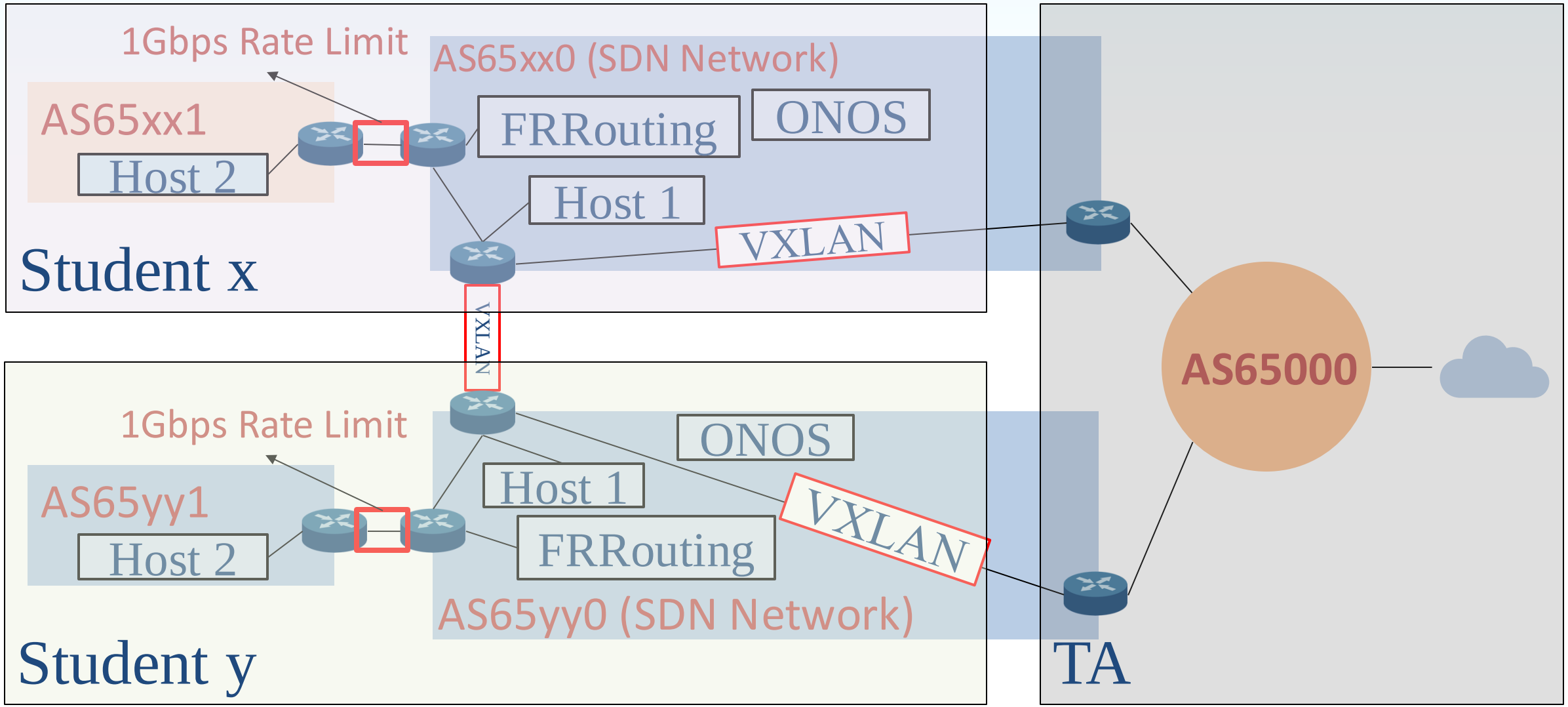
- Internet Service Provider (ISP)
  - A company or organization that provides individuals, enterprises, and other entities access to the internet.
- Key Concepts
  - Internet Access
  - Internet Service
  - IP Addresses
- If you have been assigned an IP from NYCU
  - How does the Internet world routes packets to your IP's location?
  - How does others know the way to find you in the Internet world?
- Goal: A vrouter in a small ISP and that manages internet resources.

ISP Service Requirements

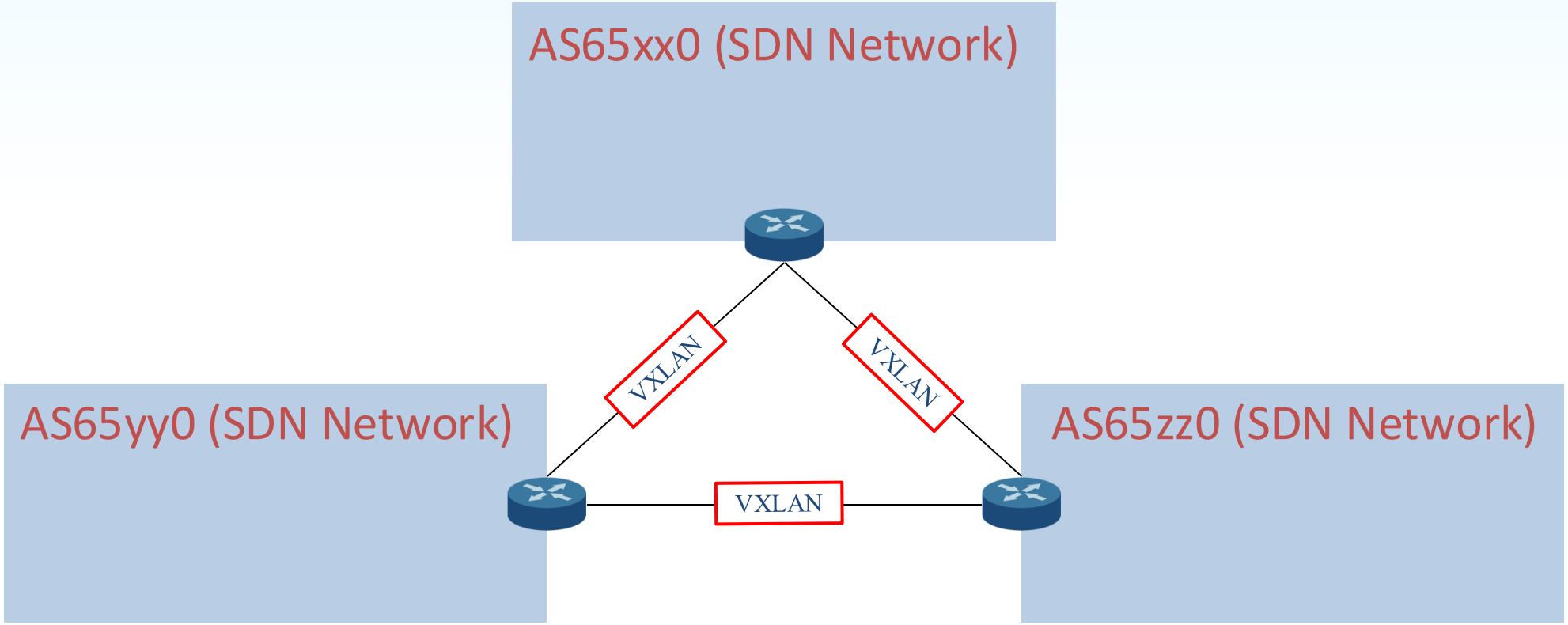
- Service requirements for customers.
  - Routers to exchange other AS's route.
  - Layer 2 modification
  - Packet handling

- Service requirements for other ISPs.
  - Packet handling
  - Quality of Service (QoS)

Topology


1Gbps Rate Limit
AS65xx0 (SDN Network)
AS65xx1
Host 2
FRRouting
ONOS
Host 1
VXLAN
Student x
VXLAN
1Gbps Rate Limit
ONOS
Host 1
AS65yy1
Host 2
FRRouting
VXLAN
AS65yy0 (SDN Network)
Student y
AS65000
TA

Topology for 3 Students



Configuration Requirements

- You are running an AS65xx0 and announcing prefixes
  - 172.16.x.0/24.
  - 2a0b:4e07:c4:xx::/64
- You help to transit prefixes announced by AS65xx1
  - 172.17.x.0/24.
  - 2a0b:4e07:c4:1xx::/64

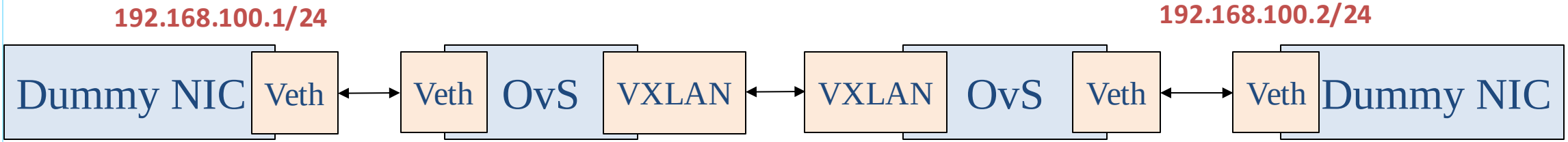
- IXP is AS65000 at 192.168.70.253/24 and fd70::fe/64
  - You have to announce the prefixes you know to the IXP
  - You can connect the IXP via
    - 192.168.70.x/24
    - fd70::x/64
  - BGP Password is winlab.nycu

OVS and VXLAN

- You can consider OVS as a bridge.
  - You can create veth pairs and connect your container
  - You can create VXLAN Tunnel

- Let's see if we can create a tunnel using `ovs-vsctl` and `ip` commands.

- TA Server have already opened a VXLAN connection with IP 192.168.100.2/24



OVS and VXLAN (cont.)

Create a ovs switch named br-ta

Set ovs protocol

Set ovs controller

```
n0ball@SDN-NFV:~/workspace$ sudo ovs-vsctl add-br br-ta -- set bridge br-ta protocols=OpenFlow14 -- set-controller br-ta tcp:192.168.100.1:6653
n0ball@SDN-NFV:~/workspace$ sudo ovs-vsctl add-port br-ta TO_TA_VXLAN -- set interface TO_TA_VXLAN type=vxlan options:remote_ip=192.168.60.200
n0ball@SDN-NFV:~/workspace$ sudo ip link add veth0 type veth peer name veth1
n0ball@SDN-NFV:~/workspace$ sudo ovs-vsctl add-port br-ta veth0
n0ball@SDN-NFV:~/workspace$ sudo ip link set veth0 up
n0ball@SDN-NFV:~/workspace$ sudo ip link set veth1 up
n0ball@SDN-NFV:~/workspace$ sudo ip address add 192.168.100.1/24 dev veth1
n0ball@SDN-NFV:~/workspace$ ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=7.59 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=64 time=5.55 ms
64 bytes from 192.168.100.2: icmp_seq=3 ttl=64 time=4.91 ms
^C
--- 192.168.100.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2053ms
rtt min/avg/max/mdev = 4.907/6.014/7.587/1.142 ms
```

Create a VXLAN port name TO_TA_VXLAN on ovs swtich br-ta

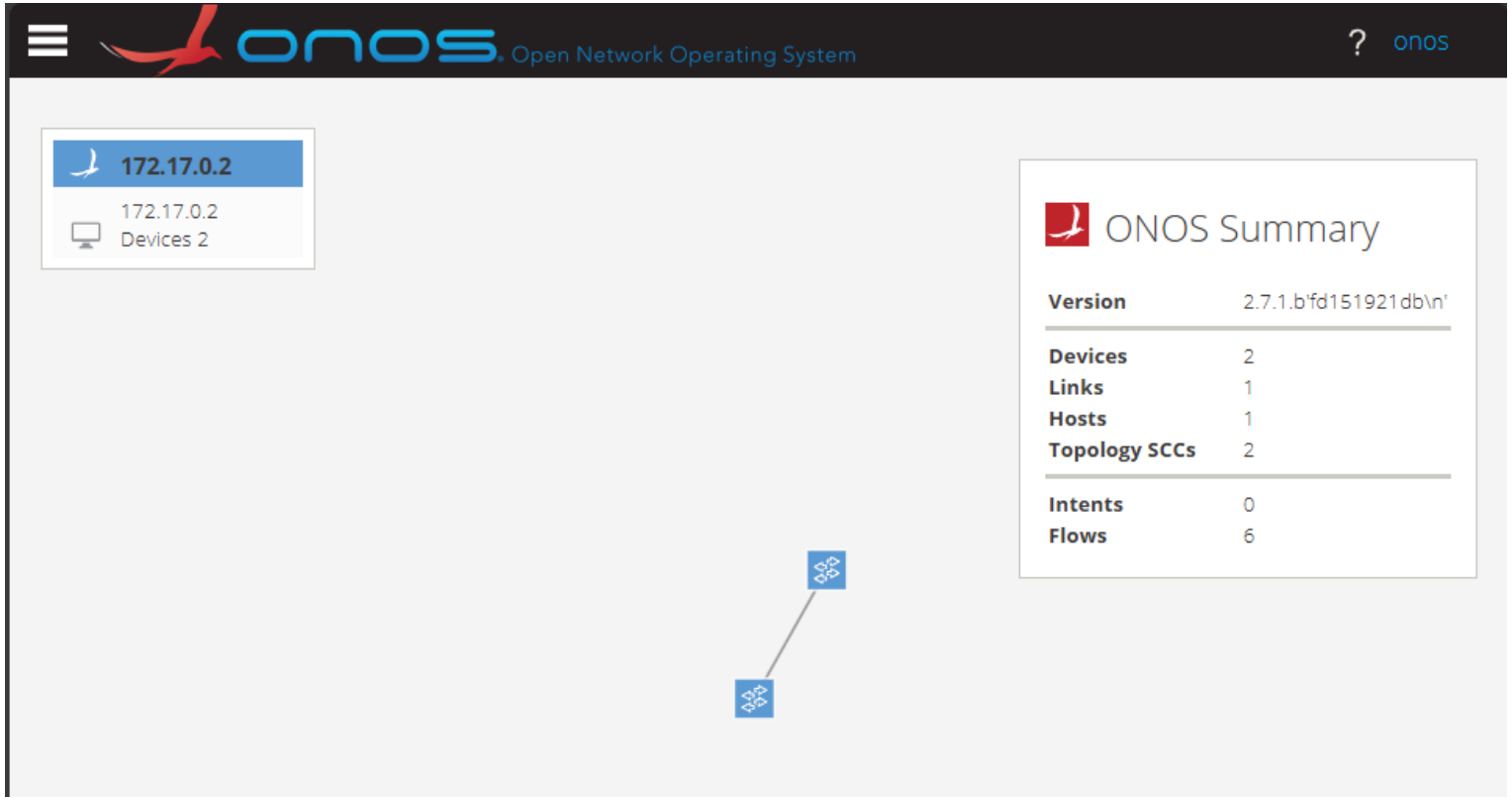
Set VXLAN connected to IP

OVS and VXLAN (cont.)

- You can use docker to create an onos controller

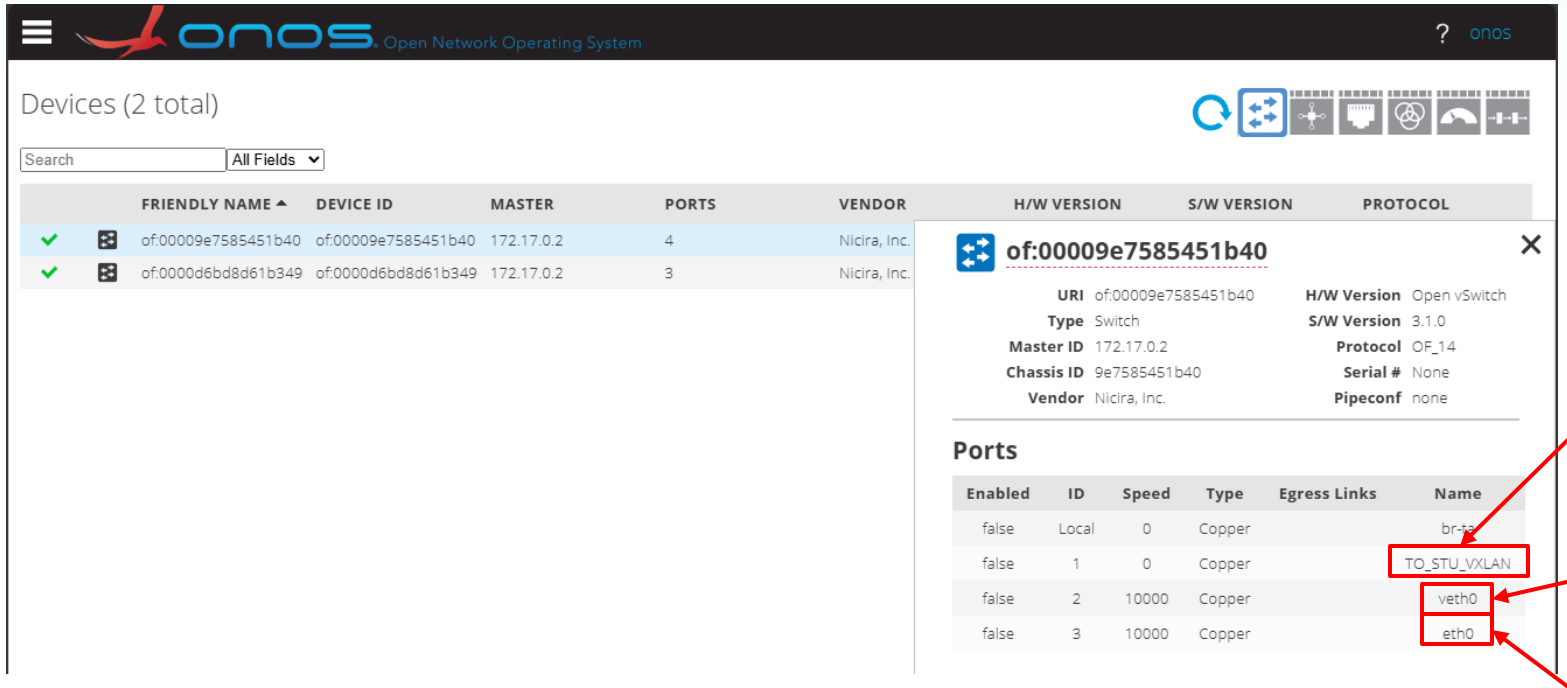
```
n0ball@SDN-NFV:~/workspace$ docker run --rm --name onos -d -p 8181:8181 -p 6653:6653 -p 8101:8101 onosproject/onos:2.7-latest
```

- If you have finish lest step, you shall see two switches connected



OVS and VXLAN (cont.)

- You can see ports on the onos GUI that TA have created for you.



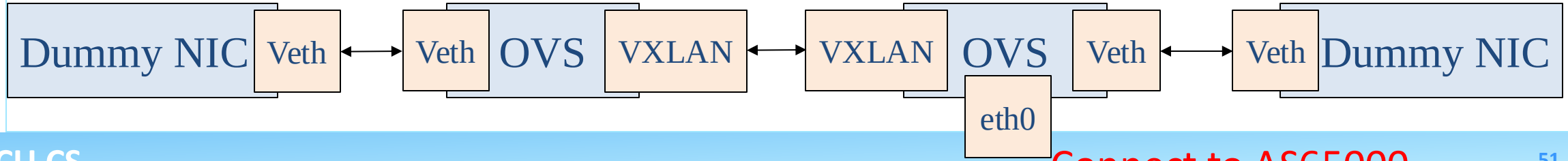
VLAN Port

A Veth for dummy NIC

Connect to AS65000

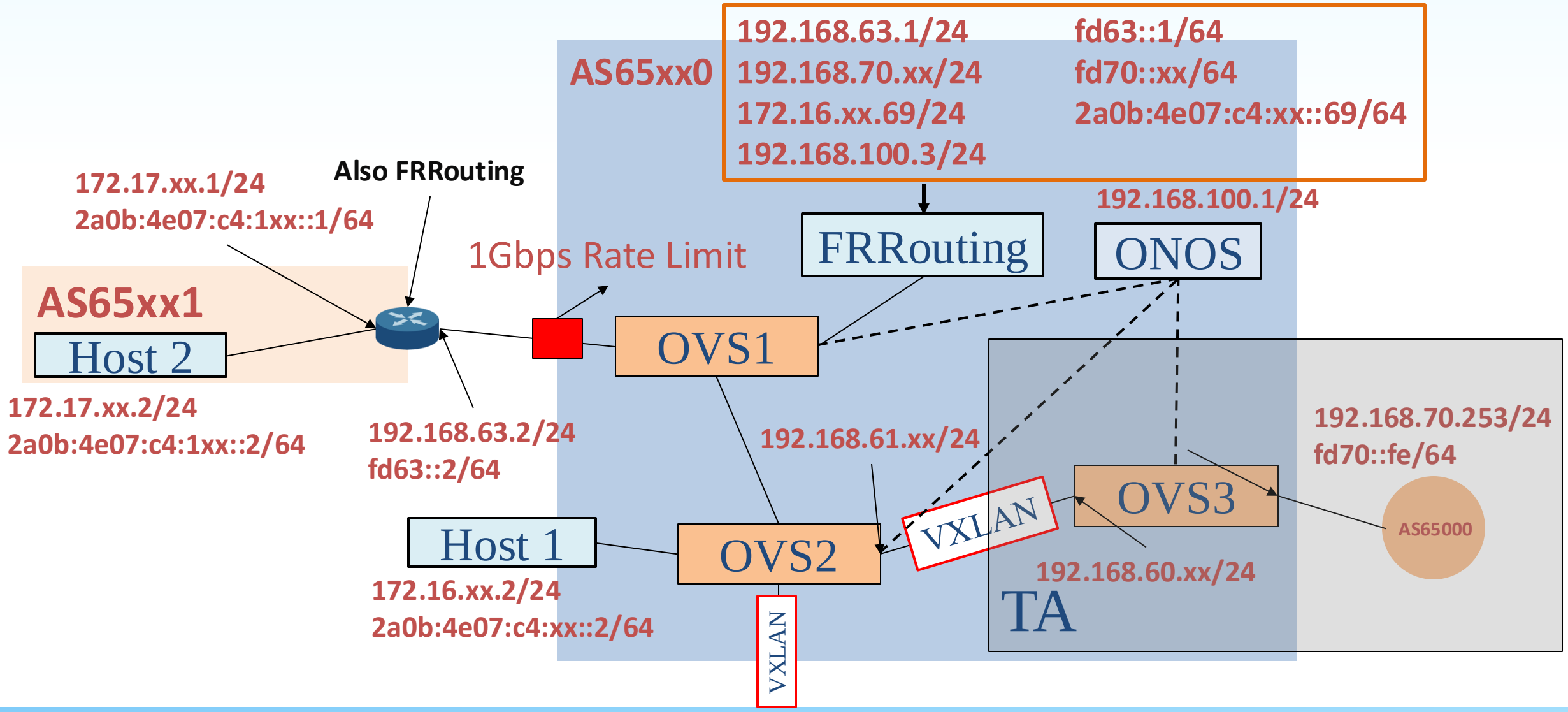
192.168.100.1/24

192.168.100.2/24

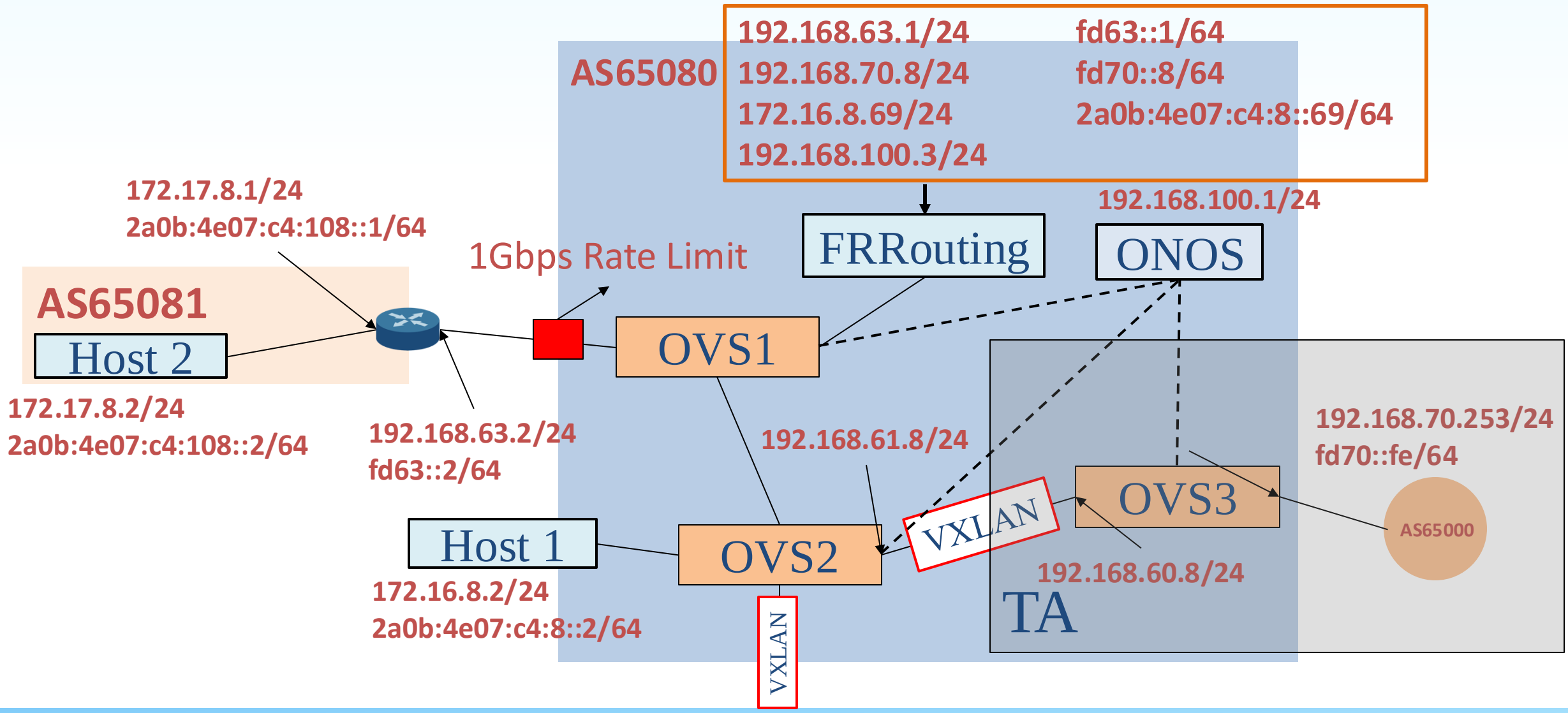


Connect to AS65000

Topology Template

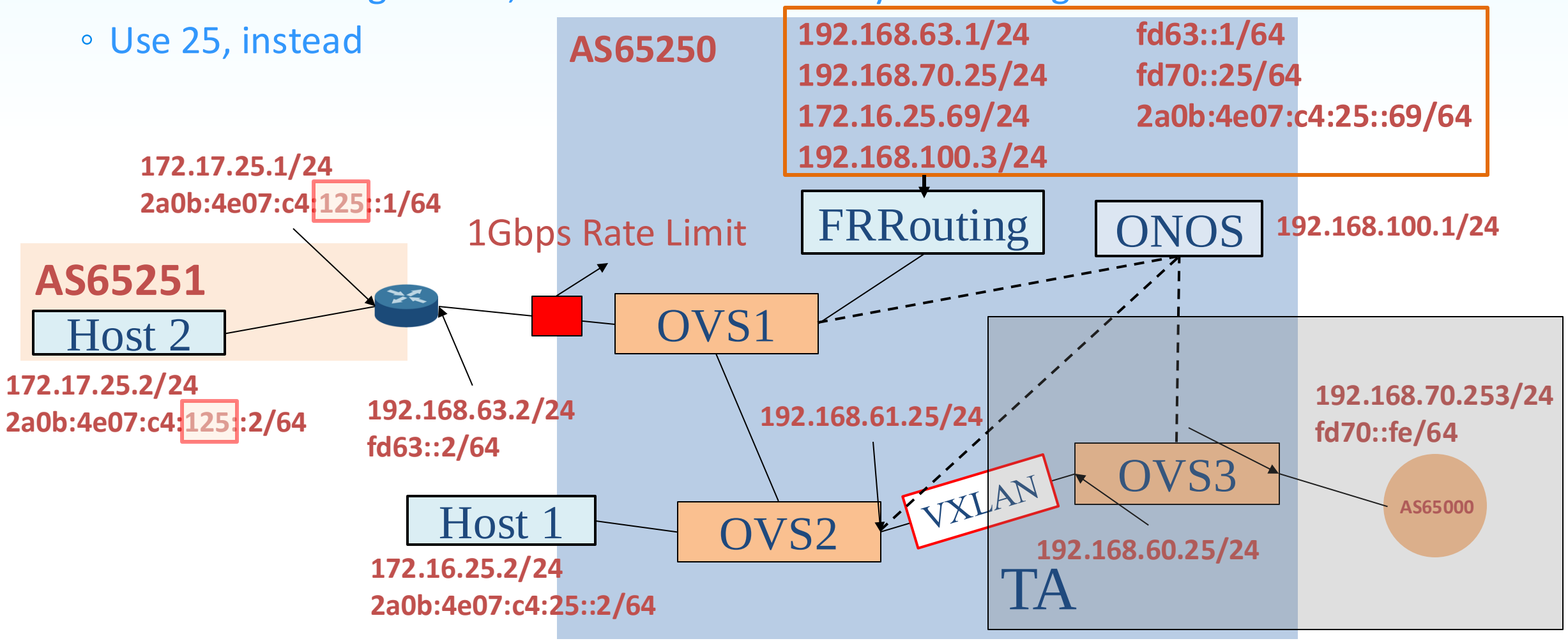


Sample Topology (x=3)



Sample Topology (x=25)

- To avoid misconfiguration, don't use a-f as IP in your settings.
  - Use 25, instead



Review of Labs

- Lab2
  - ONOS API
  - Flow rules
- Lab3
  - Mac learning
  - Proxy ARP
- Lab4
  - Intent
  - Meter Table
- Lab5 Network Function Virtualization
  - Simulate Autonomous Systems (AS)

vRouter TODO List

- Create a VXLAN tunnel between your SDN network and TA's server.
- ONOS can control OVS.
- Adding IPV6 capability.
- Intra domain packet handling.
  - Make sure everything works fine within intra domain traffic.
- Virtual Gateway function.
- Edge routers can ping each other
  - Since FRRouting is not directly connect to external router.
- FRRouting can push FIB to ONOS controller.
  - (Connection between 192.168.100.1 and 192.168.100.3)
- Deal with inter-domain traffics, i.e. ASx's hosts can ping ASy's hosts.
- Deal with transit traffics, i.e. ASx's hosts can ping from outside the world.

APP Configs

- Recommended configs

- Create a VXLAN tunnel between your SDN network and TA's server.
- ONOS can control OVS.
- Adding IPV6 capability.
- Intra/inter-domain packet handling.
- Virtual Gateway function.
- Edge routers can ping each other (FRRouting is not directly connect to external router).
- FRRouting can push FIB to ONOS controller.
- Deal with inter-domain traffics, i.e. ASx's hosts can ping ASy's hosts.
- Deal with transit traffics, i.e. ASx's hosts can ping from outside the world.

```
28      "apps": {
29          "nycu.sdnfv.vrouter": {
30              "router": {
31                  "vrrouting": "of:0000000000000001/4",
32                  "vrrouting-mac": "56:6c:11:ed:b9:28",
33                  "gateway-ip4": "172.16.1.1",
34                  "gateway-ip6": "2a0b:4e07:c4:1::1",
35                  "gateway-mac": "00:00:00:00:00:02",
36                  "v4-peers": [
37                      "192.168.70.1, 192.168.70.253",
38                      "192.168.63.1, 192.168.63.2"
39                  ],
40                  "v6-peers": [
41                      "fd70::1, fe80::42:c0ff:fea8:46fd",
42                      "fd63::1, fe80::a01d:f8ff:fea9:4d40"
43                  ]
44              }
45          },
46          "nycu.sdnfv.proxyarp": {
47              "virtual-arps": {
48                  "virtual-ip4": "172.16.1.1",
49                  "virtual-ip6": "2a0b:4e07:c4:1::1",
50                  "virtual-mac": "00:00:00:00:00:02"
51              }
52          }
53      }
54  }
```

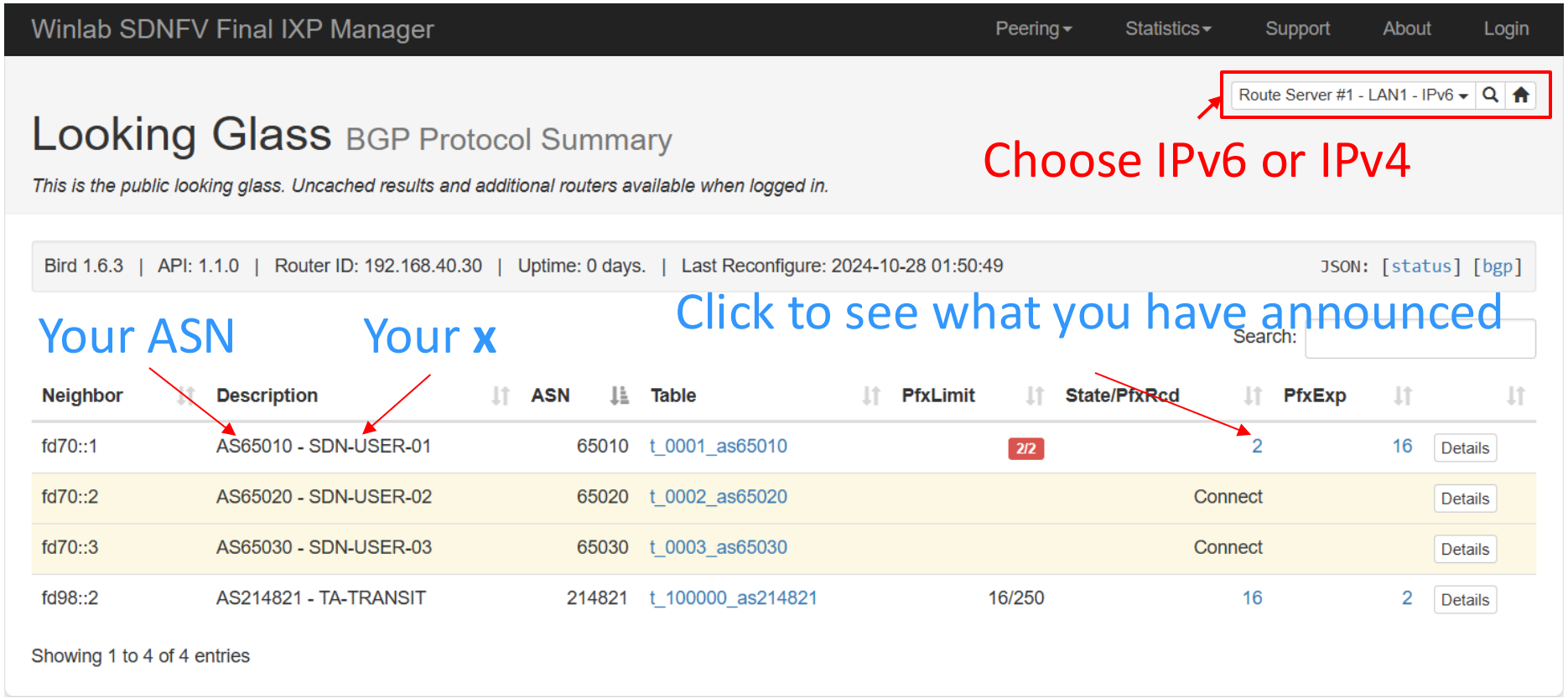
SDN AS Requirements

- For service customers (AS65xx0 SDN Network, **host 1**)
  - Able to ping **FRRouting's** IP (172.16.xx.69/24).
  - Able to ping **student y's FRRouting's** IP (172.16.yy.69/24).
  - Use https://tools.keycdn.com/ipv6-ping to see ICMP replies.

- For transit ISP (AS65xx1, **host 2**)
  - Able to ping **host 1** IP (172.16.xx.2/24).
  - Able to ping **student y's FRRouting's** IP (172.16.yy.69/24).
  - Able to ping **student y's host 2's** IP (172.17.yy.2/24).
  - Use https://tools.keycdn.com/ipv6-ping to see ICMP replies.

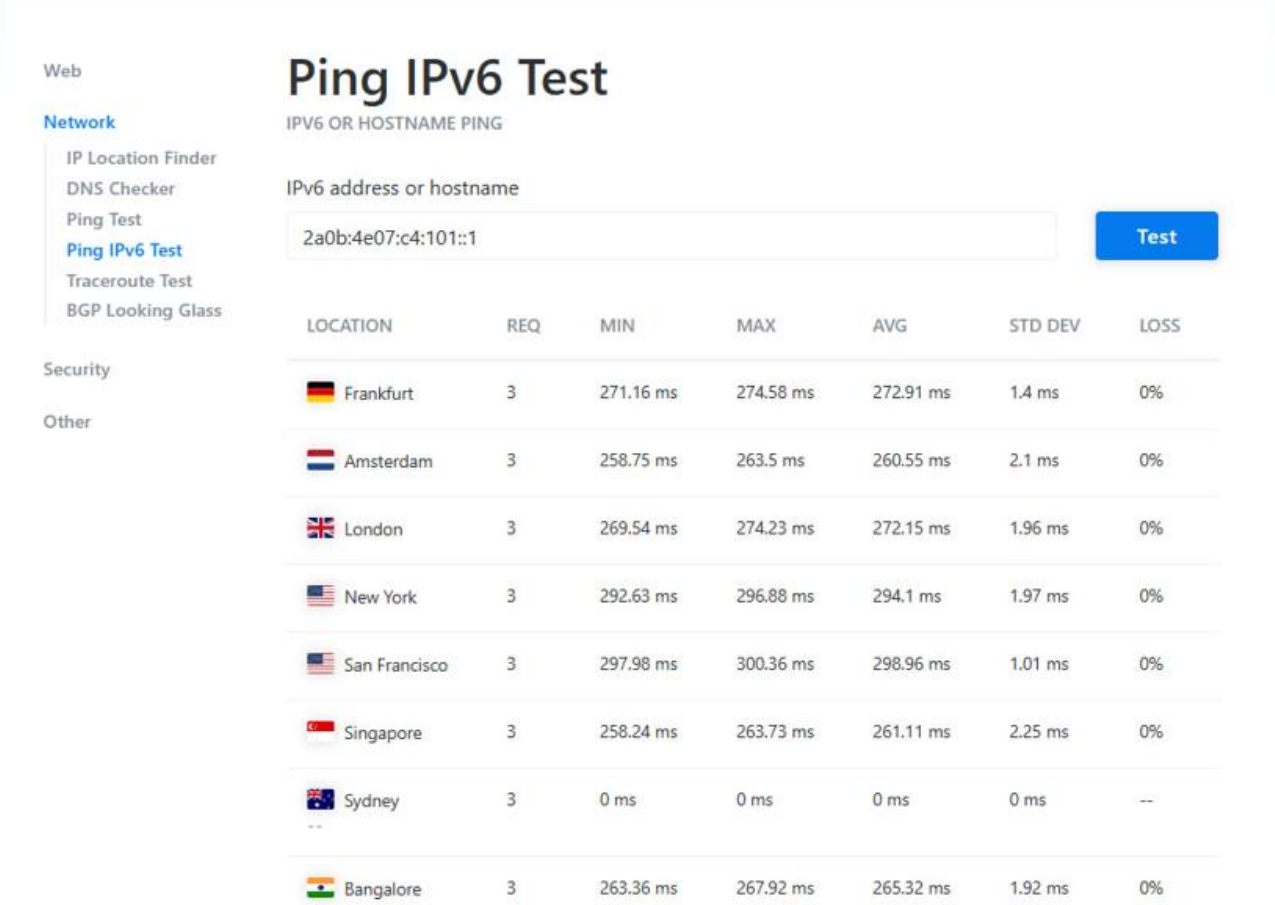
SDN Requirements

- You can see what you have announced via IXP Manager
- http://140.113.60.171:8880/lg



SDN Requirements

- You can see that you have connected to the outside world
- https://tools.keycdn.com/ipv6-ping

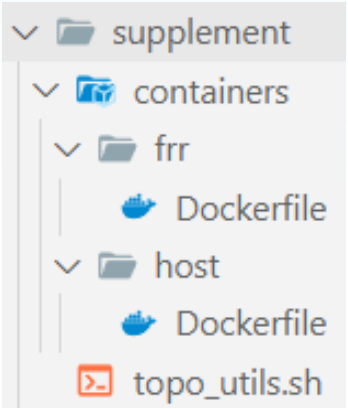


OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Supplement
- Scoring Criteria
- Reference

Helper Scripts

- There is a utils_topo.sh that might help you to create your topologies.
  - Use brctl for building linux network bridges
  - Use ovs-vsctl for building ovs bridges



- Hints
  - Remember to use your old project codes
  - ONOS might have bugs due to it has not been updated for 2 years
    - Use wireshark to make sure that the packet is exactly what you expected
    - Print debug is your best friend
    - Read the docs
  - You can turn reactive forward to test your topologies

TA Contacts

- If you have any problem
  - Mail to me
  - Register demo time for help

- Final Project
  - Team Register (should be done before 12/5, update at everyday noon)

    https://docs.google.com/spreadsheets/d/18jBtdl7BgK_GIqJtgEI210icpPq_Et1tVk8eLnFe6Kk/edit?usp=sharing
  - Demo Register

    https://calendar.google.com/calendar/u/0/appointments/AcZssZ2sbtt-446xWK_xPxxIv22bY-FV947i-odtBV4=
    - Deadline: 12/31

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Supplement
- Scoring Criteria
- Reference

Deployment Requirements

- **DO NOT TRY TO ATTACK EITHER OTHER STUDENTS OR TA SERVERS**
- You are required to create a Makefile so that you are able to
  - Clear the entire project via **make clean** command
  - Deploy the project via **make deploy** command

- **Only openflow (and route service) related apps can be use**

```
*   8 org.onosproject.drivers              2.7.1.SNAPSHOT Default Drivers
*  15 org.onosproject.fpm                  2.7.1.SNAPSHOT FIB Push Manager (FPM) Route Receiver
*  21 org.onosproject.gui2                 2.7.1.SNAPSHOT ONOS GUI2
*  36 org.onosproject.hostprovider         2.7.1.SNAPSHOT Host Location Provider
* 100 org.onosproject.lldpprovider         2.7.1.SNAPSHOT LLDP Link Provider
* 102 org.onosproject.openflow             2.7.1.SNAPSHOT OpenFlow Provider Suite
* 101 org.onosproject.openflow-base        2.7.1.SNAPSHOT OpenFlow Base Provider
*   7 org.onosproject.optical-model        2.7.1.SNAPSHOT Optical Network Model
*  14 org.onosproject.route-service        2.7.1.SNAPSHOT Route Service Server
```

- **If not, you will be scored 0**

Scores

- Intra-domain traffic (from both AS)
  - IPv4 **(6 points)**
  - IPv6 **(4 points)**
- Inter-domain traffic (from both AS)
  - IPv4 **(18 points)**
  - IPv6 **(12 points)**
- Transit traffic
  - IPv4 **(18 points)**
  - IPv6 **(12 points)**
- Routes shown in IXP Manager
  - IPv4 **(12 points)**
  - IPv6 **(8 points)**
- Able to communicate with the outside world **(10 points)**

OUTLINE

- Review of Labs
- Virtual Router Explained
- Virtual Router Specification
- ONOS App and Services in Use
- In Used App Configurations
- Virtual Router Workflow
- Supplement
- Scoring Criteria
- Reference

Reference

- RouteService (https://javadoc.io/doc/org.onosproject/onos-cli/1.8.1/org/onosproject/incubator/net/routing/package-summary.html)
- ONOS Java API (2.7.0) (https://api.onosproject.org/2.7.0/apidocs/index.html)
- ovs-vsctl(8) - Linux manual page (https://man7.org/linux/man-pages/man8/ovs-vsctl.8.html)
- ip(8) - Linux man page (https://linux.die.net/man/8/ip)