Test Readiness Review

Containerized, Intelligent Network Functions on Hosts

Project Instructor: Dr. Kevin Gifford
Project Advisor: Dr. Levi Perigo

Team 06:

Afure Martha Oyibo
Kiran Yeshwanth
Manesh Yadav
Prarthana Shedge
Soumya Velamala

Agenda

Topic	Presenter	Slide No.
Project Objectives	Soumya	3
Levels of Success	Soumya	4
FBD,CONOPS and Project Implementation	Prarthana	5 - 8
Critical Project Elements	Martha	9 - 11
Project Updates	Martha	12
Work Plan	Martha	13 - 15
Test Readiness Scope	Kiran	16 - 24
Test Readiness Status	Manesh	25 - 31
Parts and Procurements	Soumya	32 - 35
Appendix: Test Results	Prarthana	38 - 44

Project Objectives

Objective

- Create a low-cost solution for the deployment of SDN based network services such as routers, firewalls, VoIP, etc. on the host devices through NFV:
  - NFV based services will be deployed through container based VNFs
  - Enhanced performance by bringing SDN based intelligence closer to the end hosts
  - Rapid deployment of network services

Stretch Goal

- Detection of network failovers and implementation of redundant solutions:
  - Reduces the risk of packet losses
  - Seamless end user experience

Levels of Success

Level 1: Configuration Tests

- Test the deployment of VNFs
- Test the service chain creation
- Test the reachability between network devices

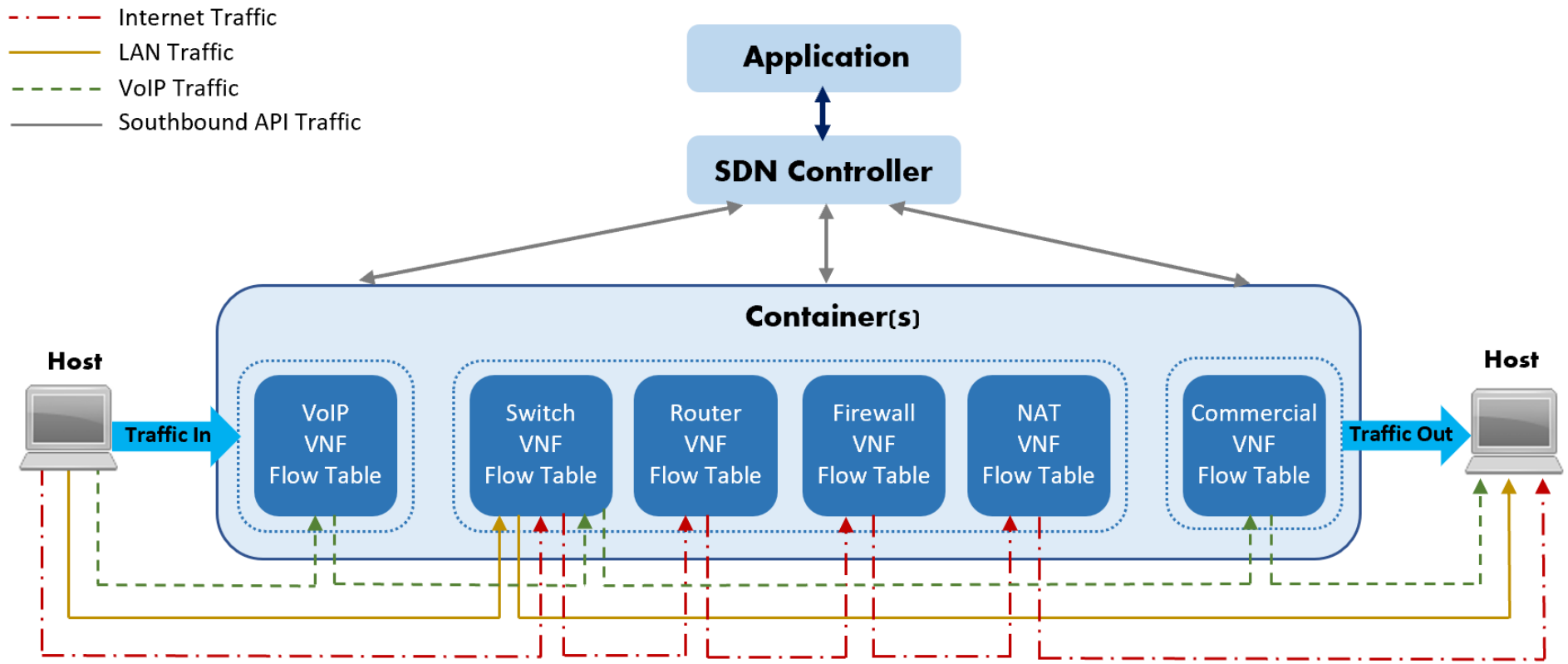
Level 2: Performance Tests

- Test network performance for different traffic types
- Distinct service types, transmission rates and packet sizes
- Evaluate QoS and throughput

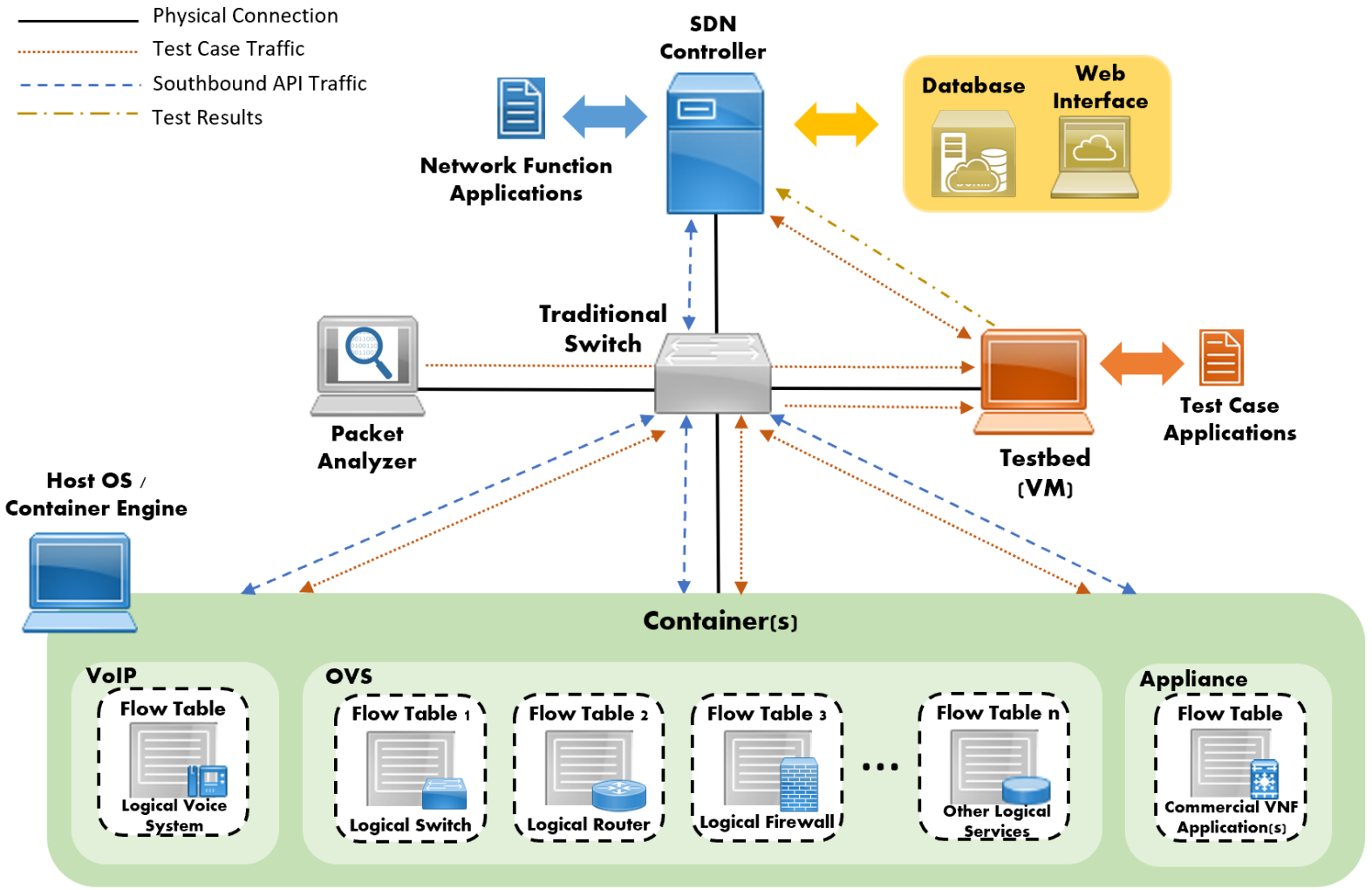
Level 3: Failover Tests

- Test the network redundancy
- Device failure, link failures, network congestion, security breach
- Test the network's ability to detect and allocate redundant resources

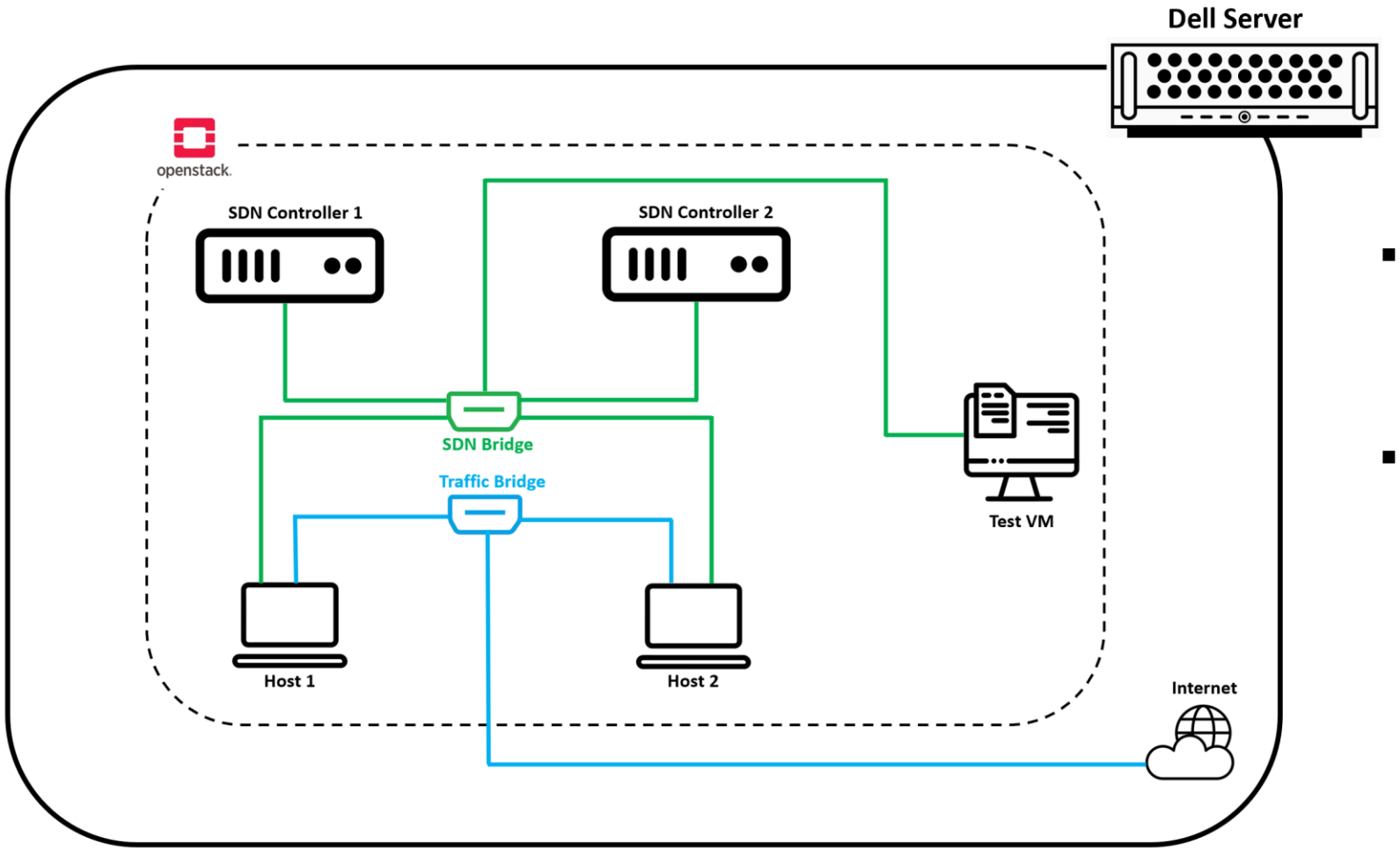
Functional Block Diagram (FBD)



Concept of Operations (CONOPS)

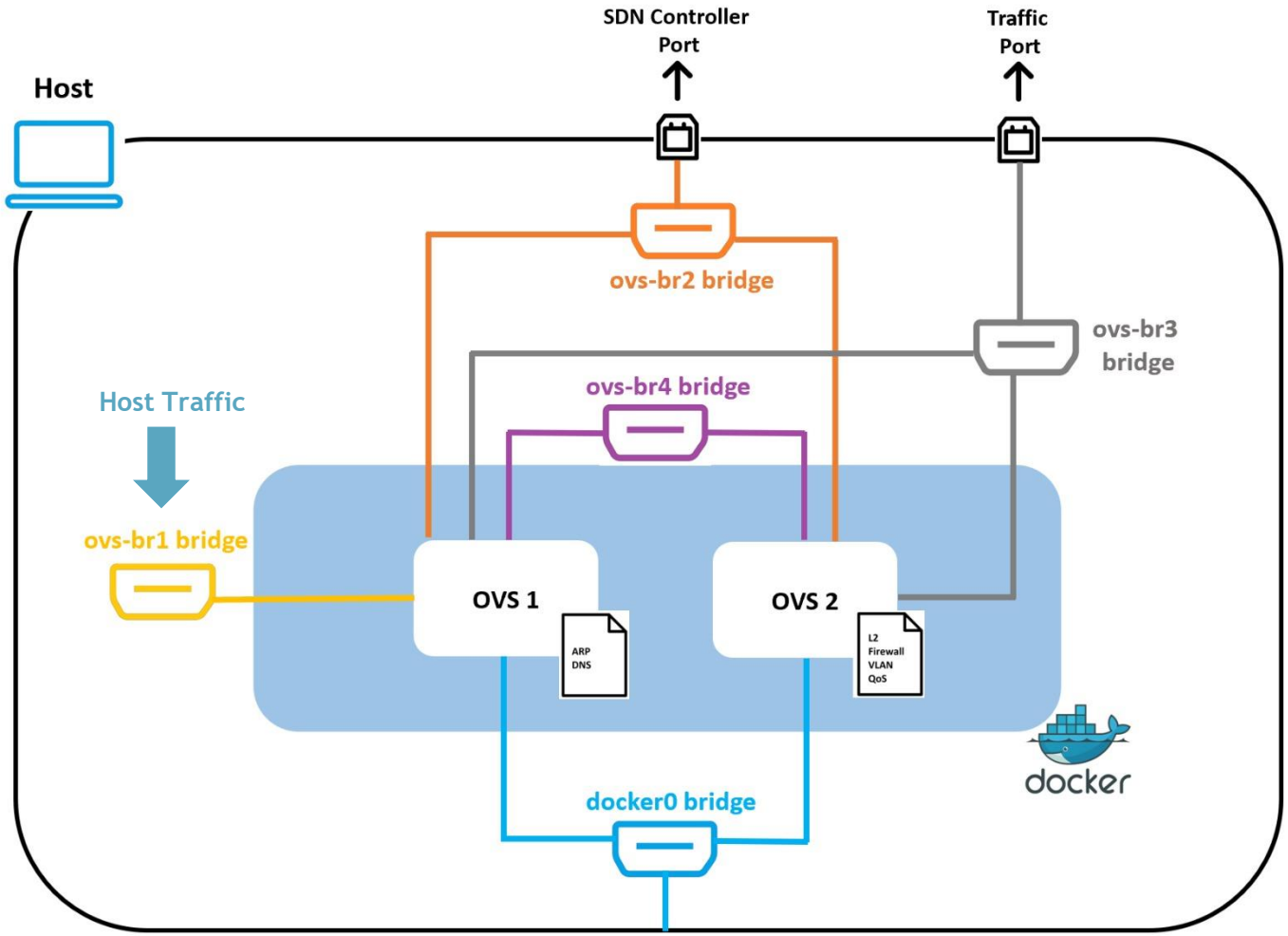


Project Implementation - Network



- Multiple VMs are initialized using Openstack which represents different network elements such as hosts, controllers and test environment
- Virtual network bridges are used to establish connectivity between network elements and the Internet

Project Implementation - Host



- Host networking consists of two OVS switches spun up using docker containers
- Multiple Linux bridges are used to establish network connectivity between host and network functions and elements through OVS switches
- All host traffic is directed to ovs-br1 bridge using the *default ip routes*

Critical Project Elements - Technical

CPE 1.1: VNF Creation

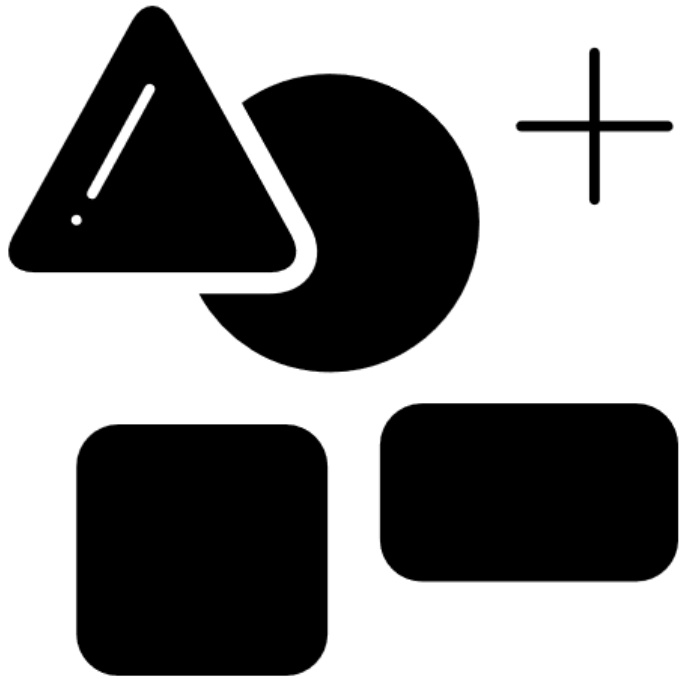
- Deploy OVS Switch using containers
- Create VNFs corresponding to network services on OVS Switches

CPE 1.2: Configure SDN Infrastructure

- Deploy SDN Controller to configure and manage flows in VNFs
- Deploy overlay network for connectivity
- Deploy packet analyzer for analysis

CPE 1.3: Test VM Creation

- Deploy Linux based Test VM
- Emulate test environment on Test VM
- Perform operational, performance and configuration tests



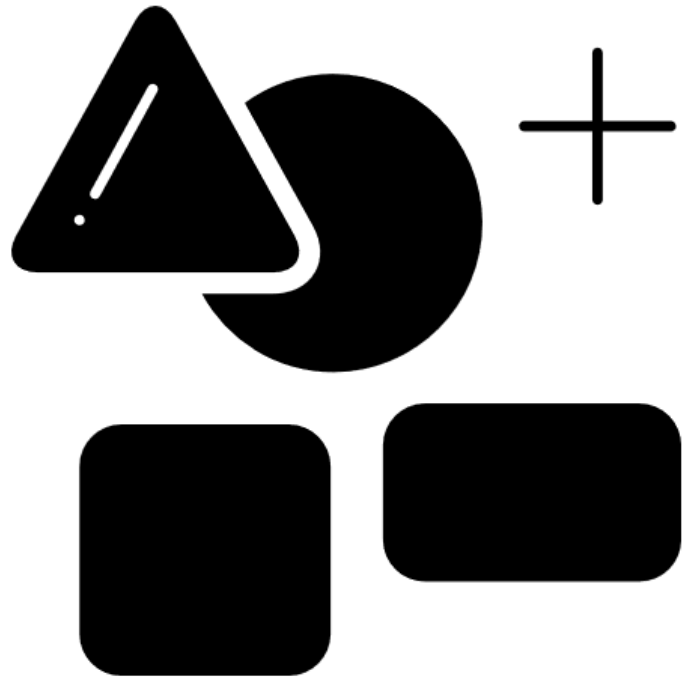
Critical Project Elements – Technical (continued)

CPE 1.4: Service Chain Creation

- Create multiple service chains
- Simulate and test different traffic scenarios

CPE 1.5: Test results (Storage and display)

- Database to store and parse the output of test results
- Web-interface to display the output of test results



Critical Project Elements - Logistical

CPE 2.1: Hardware Devices

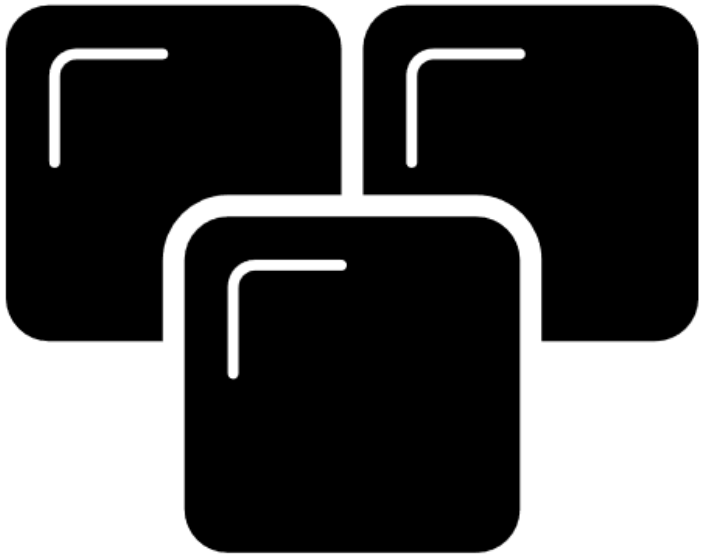
- Using servers and traditional switches available in Telecommunications lab
- Laptops with Windows, Linux and MacOS as host devices

CPE 2.2: Containers/ Controllers

- Using open-source software for topology creation
- Dockers as containers, ONOS as SDN Controller

CPE 2.3: Knowledge and concepts

- Understand the functionality of containers
- Determine web GUI framework and learn the functionality

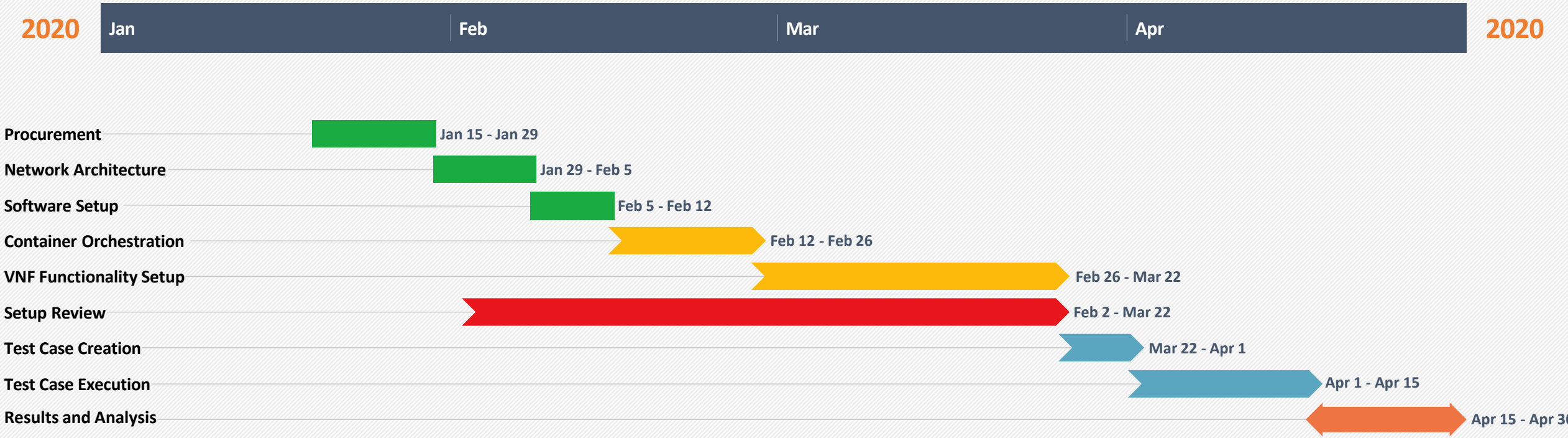


Project Updates

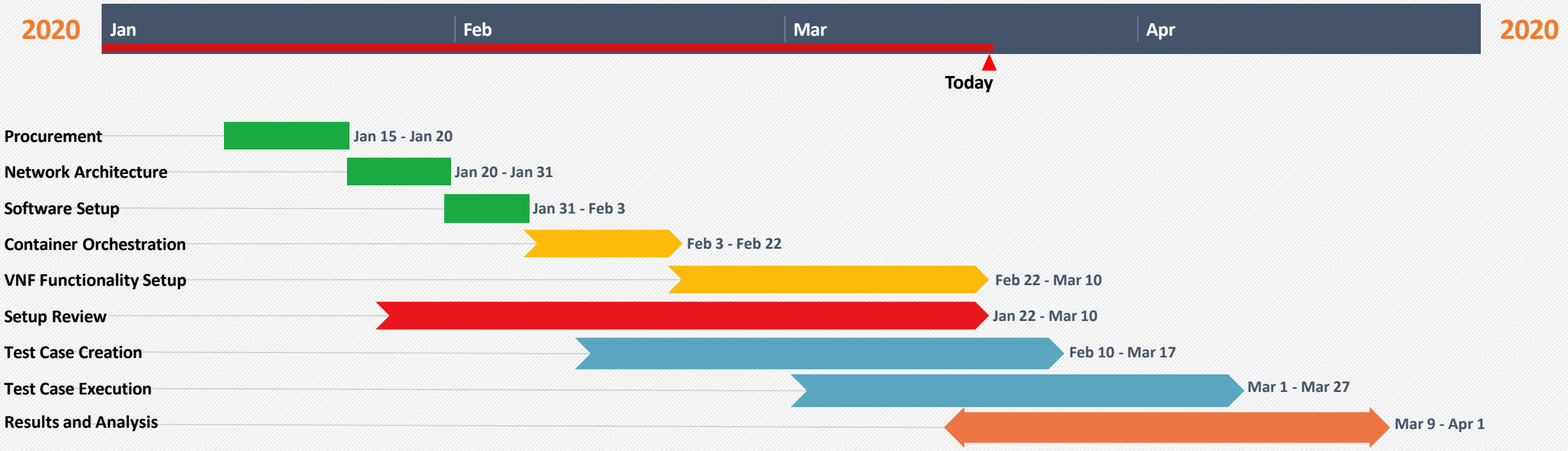
- Instead of configuring 2 Dell Servers, we now have implemented all the network elements on a single server using virtualization with KVM/Qemu2 managed by OpenStack.
- For code management of test cases, we now would use GitHub and Jenkins
- After having discussed with our academic advisor, RYU is now configured as our SDN controller, due to python programmability.

WORKPLAN

WORK PLAN - PREVIOUS



WORK PLAN - CURRENT



Test Readiness - Scope

LEVEL 6 Multi-environment Testing

LEVEL 5 Stress Testing

LEVEL 4 Performance Testing

LEVEL 3 Compatibility Testing

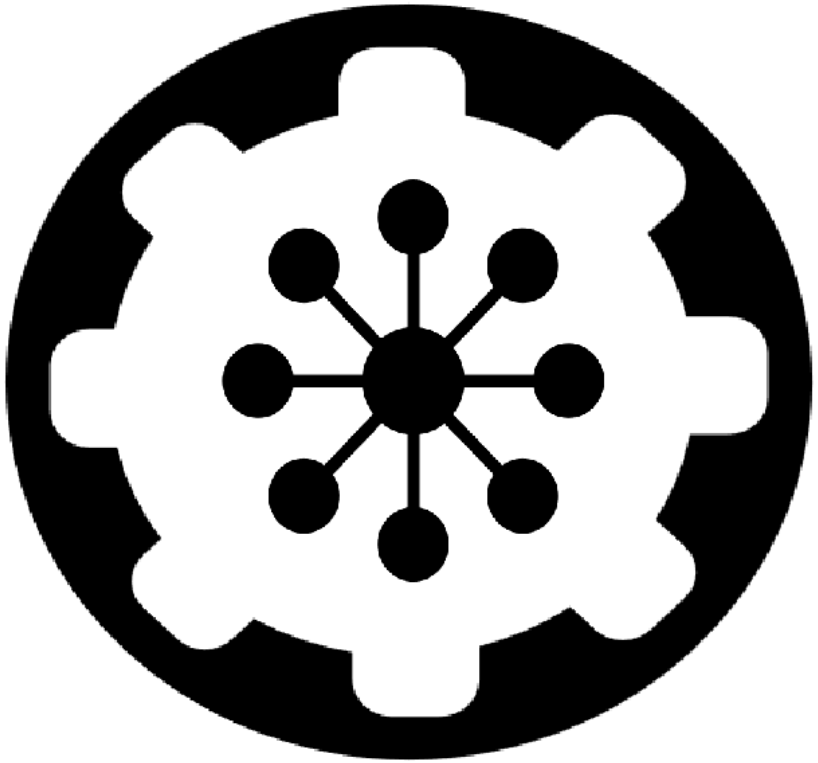
LEVEL 2 Integration Testing

LEVEL 1 Deployment Testing

Level 1: Deployment of Network Infrastructure

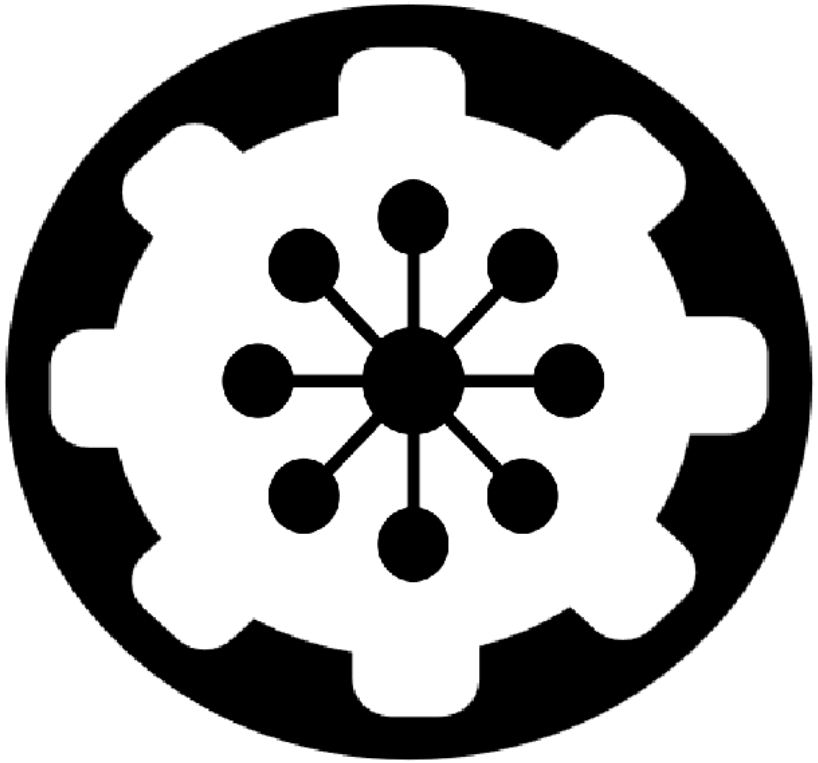
Test the deployment of network devices & software tools

Openstack Orchestration	• Verify the creation and connectivity of VMs (Network Elements) through OpenStack
Containers	• Verify the installation of OVS switches on the host devices using containers. Validate the deployment and configuration of VNFs on OVS switches
SDN Controller	• Validate the deployment and configuration of containers in the Controller VMs. Verify installation of SDN Controller application in the Controller VMs



Level 1: Deployment of Network Infrastructure (continued)

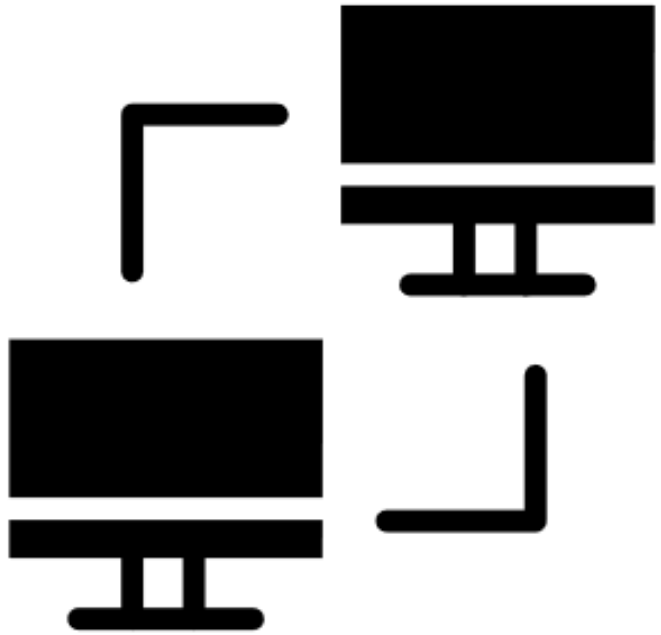
Storage & Output	• Validate deployment of storage instances and web interface
Test VM	• Verify the creation of test environment and installation of traffic generator and packet analyzer in Test VM
Traditional Devices	• Validate deployment of intermediate network devices using python script



Level 2: Integration of Network Infrastructure

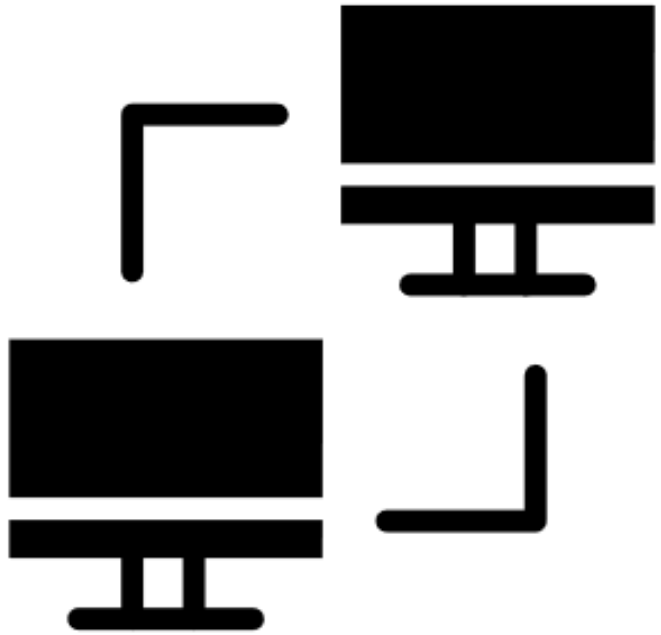
Test the reachability between network devices & VNFs

Containers	• Verify the configuration, interface status of the OVS switches. Validate the creation of service chaining between different VNFs
SDN Controller	• Ensure the reachability between the Controller VMs and the intermediate network devices. Verify the connectivity from Controller VMs to SDN Controller application, storage instances and web interfaces



Level 2: Integration of Network Infrastructure (continued)

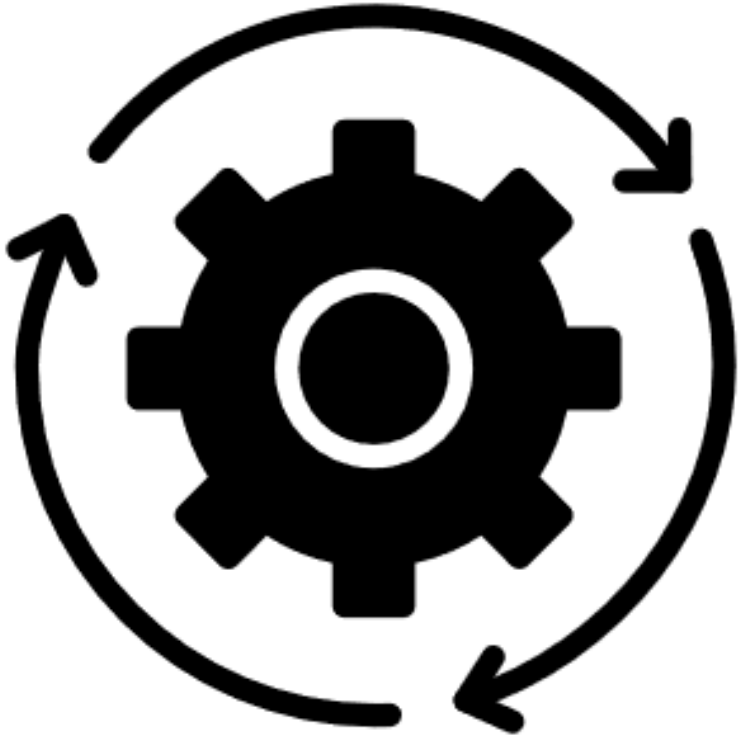
Storage & Output	• Validate the connectivity between storage devices and the web interfaces
Test VM	• Perform the health check of the Test VM interfaces and ensure the connectivity of the TEST VM to the intermediate devices
Traditional Devices	• Verify the status of interfaces connected to the SDN infrastructure devices



Level 3: Compatibility between Network Devices

Verify the seamless functionality of network protocols between network devices

Containers	• Verify installation of flow entries in multiple flow tables in the OVS switches
SDN Controller	• Validate the NBI operation between Controller VM and application layer, the SBI operation between Controller VM and network devices
Storage & Output	• Verify the storage and display of results using mock input values. HTTP Test Automation Hooks to check the web interface functionality



Level 4: Performance Test for VNFs

Test the functionality of VNFs on host devices

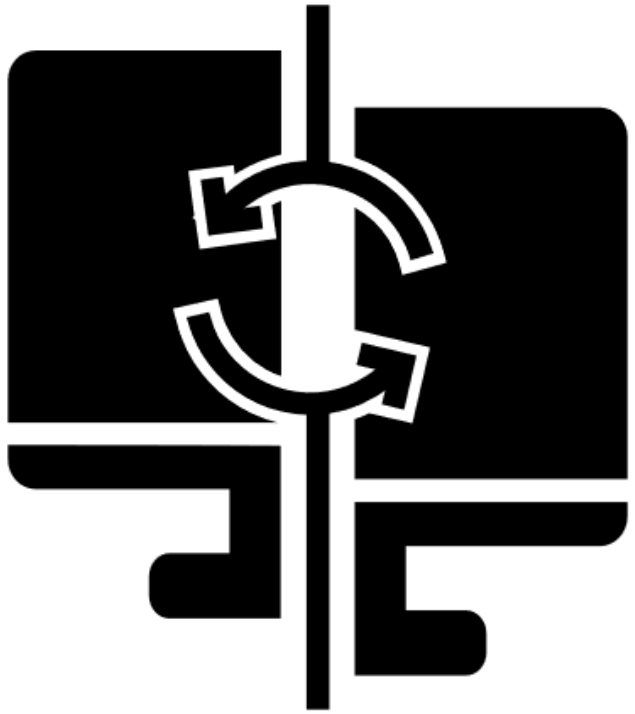
Containers

- Verify the functionality of VNFs with respect to expected output
- Evaluate the performance based on latency, throughput, utilization
- Multiple traffic scenarios generated through traffic generator will be used for testing

Level 5: Stress Test for Network Devices

Test the backup scenarios in case of failovers

SDN Controllers	• Verify the seamless functionality provided by the secondary SDN Controller in case the primary controller fails
Traditional Devices	• Verify the reachability of the host devices and Test VM with the SDN Controller through the traffic link in case the virtual bridge goes down



Level 6: Multi-environment Network Deployment

Test the deployment of network services and functionality on different Operating Systems

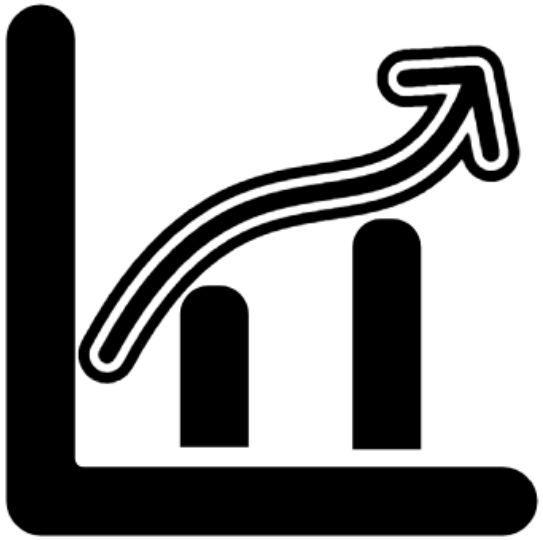
Containers

- Deploy network services using containers on different OS (Windows, Linux, MacOS) and test the functionality on different environment

SDN Controller

- Deploy SDN Controller using VMs and bare metal servers and test the functionality on different environment

Test Readiness – Procedure & Status



LEVEL 6 Multi-environment Testing

LEVEL 5 Stress Testing

LEVEL 4 Performance Testing

LEVEL 3 Compatibility Testing

LEVEL 2 Integration Testing

LEVEL 1 Deployment Testing

Level 1: Deployment of Network Infrastructure

Verify setup & functionality of hardware devices & underlying software elements

1. Check if OpenStack is deployed successfully by SSH/VPN login into the Dell Server
2. In OpenStack check if VMs are successfully created and running either using OpenStack GUI or CLI
3. Check if created VMs have IP address allocated on ports
4. Check if Docker and Wireshark are successfully installed on all VMs by SSH/VPN login into the VMs
5. Check for successful installation and run status of RYU controller, OVS switches and traffic generator on Docker containers using Docker CLI
6. Check creation of virtual network bridges for management and data communication
7. Verify successful creation and deployment of flowtables corresponding to respective VNFs on OVS switches

Status: PASS

Level 2: Integration of Network Infrastructure

Verifying connectivity between network elements

1. Ping VMs to check successful intra domain and internet connectivity
2. Verify successful bridging between container ports, VM interfaces and Linux bridges to support multiple service chaining
3. Use ping to check connectivity between OVS switches and SDN controller
4. Monitor continuous Up-status of container and VM interfaces using CLI

Status: PASS

Level 3: Compatibility between Network Devices

Verifying protocol level connectivity between network elements

1. Using OVS CLI, verify OpenFlow connection between OVS switches and SDN controllers
2. Verify REST API connectivity for deployment of proactive flows between SDN controllers and application software by checking count of flow-entries
3. Use pull and push from GitHub and Jenkins to check connectivity between local repo and our testing framework and source-code management application

Status: ONGOING

Level 4: Performance Test for VNFs

Network performance testing of hosts

1. Use ping to verify intra and inter domain connectivity between host devices
2. Use SSH & HTTP services to check successful functionality of Firewall VNF
3. Check creation of QoS services on OVS2 switch interfaces
4. Check for implementation of QoS service on OVS2 switch for dedicated IP address
5. Verify successful domain division of IP addresses according to VLANs on OVS2 switch
6. Verify DNS VNF functionality using ping and HTTP services for checking reachability to good URLs and blocking reachability for bad URLs
7. Verify successful dynamic mapping of IP and MAC addressing in the SDN controller using Scapy

Status: PASS

Level 5: Stress Test for Network Devices

Verifying redundancy mechanism in case of failovers

1. Check for SDN controller container redundancy status using Docker Swarm
2. Verify redundant deployment of all VNF services on one OVS switch in case other OVS goes down. This is done by checking port status of OVS switches and *docker running* status

Status: PASS

Level 6: Multi-environment Network Deployment

Verifying multi-environment VNF deployment

1. Verify creation of OVS switches on different OS devices and check interpretability between these devices and network elements by following test levels 1-5

Status: ONGOING

PARTS & PROCUREMENT

LAPTOP

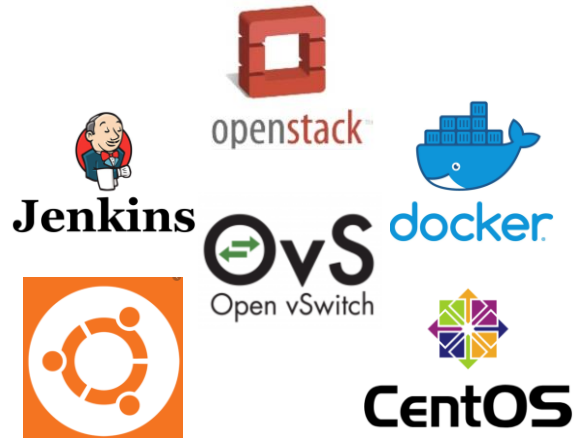
PROCESSOR	2.2 GHz Intel Core **i7**
Operating System	Mac-OS
Storage	500GB
Memory	8GB
Availability	**Procured – CU Surplus**

SERVER – FEATURES AND SPECIFICATIONS

PROCESSOR	Intel® Xeon® processor E5-2600 v4 product family
Operating System	Windows, Linux and Vmware ESX
Storage	500GB
Memory	64GB
Availability	Telecommunications Lab

SOFTWARE & IMAGES

Entity	DESCRIPTION	PRICE
OpenStack	Cloud Orchestrator/ VM Manager	Opensource
Docker Swarm	Container Manager	Opensource
Jenkins	Automation Server	Opensource
Ubuntu 16.04	Operating System	Opensource
Centos7	Operating System	Opensource
OvS	OF enabled Virtual MLS	Opensource



Q&A

Backup Slides

TEST RESULTS

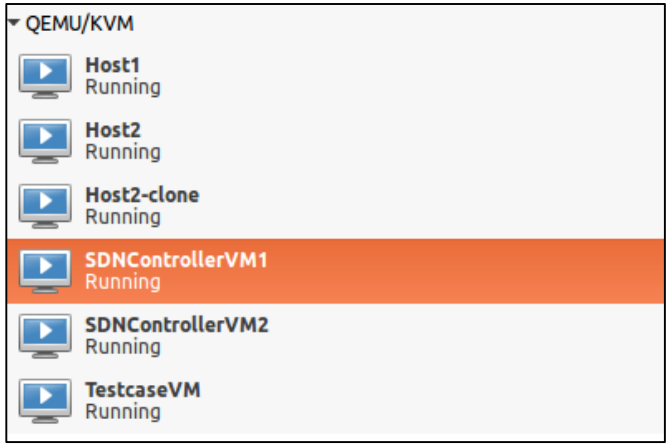
Level 1: Deployment of Network Infrastructure

1. VM creation and configuration

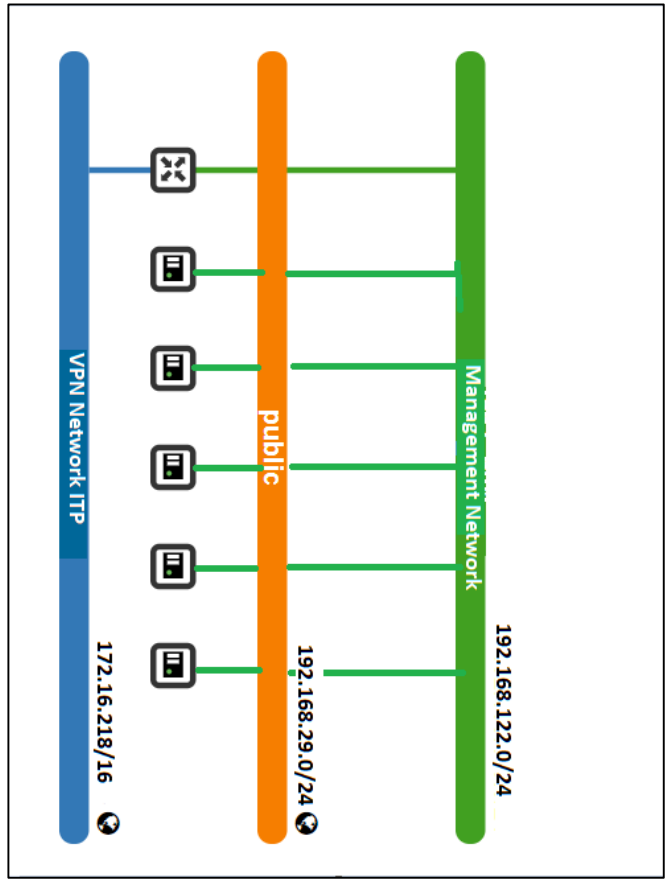
```
root@openstack ~(keystone_admin)]# openstack server list
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| ID                 | Name                 | Status | Task State | Power State | Networks          | Image Name        |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| 8b67821b-4571...   | SDNController1       | ACTIVE | None       | Running     | private=192.168.122.244 | centos      |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| 94b5783e-8278...   | SDNController2       | ACTIVE | None       | Running     | private=192.168.122.117 | centos      |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| 7a1234d5-7385...   | Host1                | ACTIVE | None       | Running     | private=192.168.122.38  | centos      |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| 8a99547e-0145...   | Host2                | ACTIVE | None       | Running     | private=192.168.122.54  | centos      |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
| 1a45c78e-7612...   | Test VM              | ACTIVE | None       | Running     | private=192.168.122.142 | centos      |
+--------------------+----------------------+--------+------------+-------------+-------------------+-------------------+
```

2. Connectivity between VMs

```
[khir@host1 ~]$ ping 192.168.122.244
PING 192.168.122.244 (192.168.122.244) 56(84) bytes of data.
64 bytes from 192.168.122.244: icmp_seq=1 ttl=64 time=0.606 ms
^C
--- 192.168.122.244 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.606/0.606/0.606/0.000 ms
[khir@host1 ~]$ ping 192.168.122.117
PING 192.168.122.117 (192.168.122.117) 56(84) bytes of data.
64 bytes from 192.168.122.117: icmp_seq=1 ttl=64 time=0.855 ms
^C
--- 192.168.122.117 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.855/0.855/0.855/0.000 ms
```

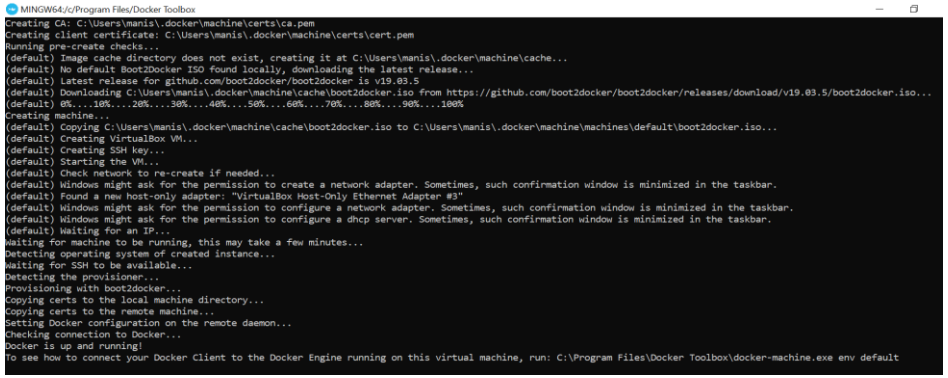


3. Connectivity to internet



Level 1: Deployment of Network Infrastructure

1. Docker installation



2. SDN Controller installation- Docker

```
[root@localhost khir]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ryu312              latest              dc4d90f61d81        3 weeks ago         293MB
ubuntu              latest              ccc6e87d482b        7 weeks ago         64.2MB
ubuntu              14.04               6e4f1fe62ff1        2 months ago        197MB
osrg/ryu            latest              ab10e91ba3b6        11 months ago       294MB
```

```
[root@localhost khir]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
37f12bc59c69        ryu312:latest       "sleep infinity"    3 weeks ago         Up 3 weeks          6633/tcp, 6653/tcp
[root@localhost khir]#
```

3. Docker version

```
[root@host1 khir]# docker --version
Docker version 19.03.7, build 7141c199a2
[root@host1 khir]#
```

4. OVS Switch installation- Docker

```
[root@host1 khir]# docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
globocom/openvswitch    latest              58215e48af7c        24 months ago       73.4MB
[root@host1 khir]# docker ps -a
CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS               NAMES
cda756561278        globocom/openvswitch    "/bin/sh -c /usr/bin…"   25 hours ago        Up 25 hours         22/tcp              vswitch2
dd1119acc593        globocom/openvswitch    "/bin/sh -c /usr/bin…"   25 hours ago        Up 25 hours         22/tcp              vswitch1
```

Level 2: Integration of Network Infrastructure

1. Virtual Network bridge creation

```
khir@eneto-compute-PowerEdge-R430:~$ virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
 virbr1               active     yes           yes
 virbr2               active     yes           yes
```

2. Bridge configurations

```
khir@eneto-compute-PowerEdge-R430:~$ virsh net-dumpxml virbr1
<network connections='3'>
  <name>virbr1</name>
  <uuid>7ccc2ac5-56c5-460c-9aab-60127c34b67b</uuid>
  <bridge name='virbr1' stp='on' delay='0'/>
  <mac address='44:34:00:34:34:34'/>
  <ip address='192.168.29.7' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.29.8' end='192.168.29.254'/>
    </dhcp>
  </ip>
</network>
```

```
khir@eneto-compute-PowerEdge-R430:~$ virsh net-dumpxml virbr2
<network connections='3'>
  <name>virbr2</name>
  <uuid>7aaa2ac5-56a5-460a-9aab-60127c34b67b</uuid>
  <bridge name='virbr2' stp='on' delay='0'/>
  <mac address='54:34:10:35:44:34'/>
</network>
```

3. OVS related bridges

```
[root@host1 khir]# ovs-vsctl show
8bf09e78-f3c6-4b58-9026-5aa431549303
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "a6662c07e6944_l"
            Interface "a6662c07e6944_l"
    Bridge br-sdn
        Port "63dadc137f5f4_l"
            Interface "63dadc137f5f4_l"
        Port "c6c6beffeebf4_l"
            Interface "c6c6beffeebf4_l"
        Port br-sdn
            Interface br-sdn
                type: internal
```

```
    Bridge br-traffic
        Port "ens10"
            Interface "ens10"
        Port br-traffic
            Interface br-traffic
                type: internal
        Port "2662d94ab5384_l"
            Interface "2662d94ab5384_l"
        Port "ab1a8567f6c14_l"
            Interface "ab1a8567f6c14_l"
    Bridge br-conn
        Port br-conn
            Interface br-conn
                type: internal
        Port "90abef82e90e4_l"
            Interface "90abef82e90e4_l"
        Port "2f4b82f11c6e4_l"
            Interface "2f4b82f11c6e4_l"
    ovs_version: "2.3.1"
```

Level 3: VNF Creation OVS Switch - Docker

1. Connectivity between SDN Controller and OVS

```
96b7266c-a535-41a1-8c45-98491245bdc6
    Bridge br-data
            Controller "tcp:192.168:122.244:6653"
                Controller "tcp:192.168:122.117:6653"
        fail_mode: secure
        Port "eth2"
            Interface "eth2"
        Port "eth1"
            Interface "eth1"
        Port br-data
            Interface br-data
                type: internal
```

2. VNF Creation Firewall

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep table=1,
 cookie=0x110, duration=61107.962s, table=1, n_packets=0, n_bytes=0, priority=151,ip,in_port=1,nw_src=192.168.29.55 actions=goto_table:2
 cookie=0xd0, duration=61107.907s, table=1, n_packets=0, n_bytes=0, priority=109,arp,in_port=1,arp_spa=192.168.29.99 actions=drop
 cookie=0xd1, duration=61107.899s, table=1, n_packets=0, n_bytes=0, priority=110,ip,in_port=1,nw_src=192.168.29.99 actions=drop
 cookie=0x115, duration=61107.868s, table=1, n_packets=0, n_bytes=0, priority=10,ip,in_port=1,nw_src=192.168.29.68 actions=goto_table:2
 cookie=0x116, duration=61107.861s, table=1, n_packets=0, n_bytes=0, priority=10,ip,in_port=1,nw_src=192.168.122.38 actions=goto_table:2
 cookie=0x11a, duration=60560.695s, table=1, n_packets=0, n_bytes=0, priority=153,arp,in_port=1,arp_spa=192.168.29.55 actions=goto_table:2
 cookie=0xcd, duration=61107.982s, table=1, n_packets=3, n_bytes=306, priority=114,ip,in_port=2,nw_dst=192.168.29.68 actions=goto_table:5
 cookie=0xd6, duration=61107.975s, table=1, n_packets=4, n_bytes=240, priority=115,arp,in_port=2,arp_tpa=192.168.29.68 actions=goto_table:5
 cookie=0x10f, duration=61107.969s, table=1, n_packets=0, n_bytes=0, priority=150,ip,in_port=1,nw_dst=192.168.29.55 actions=goto_table:2
 cookie=0xf5, duration=61107.937s, table=1, n_packets=0, n_bytes=0, priority=144,ip,in_port=2,nw_dst=192.168.122.38 actions=output:1
 cookie=0xf4, duration=61107.930s, table=1, n_packets=0, n_bytes=0, priority=145,arp,in_port=2,arp_tpa=192.168.122.38 actions=output:1
 cookie=0xce, duration=61107.922s, table=1, n_packets=0, n_bytes=0, priority=107,arp,in_port=1,arp_tpa=192.168.29.99 actions=drop
 cookie=0xcf, duration=61107.914s, table=1, n_packets=0, n_bytes=0, priority=108,ip,in_port=1,nw_dst=192.168.29.99 actions=drop
 cookie=0xce, duration=61107.876s, table=1, n_packets=0, n_bytes=0, priority=130,ip,in_port=1,nw_dst=4.4.4.4 actions=drop
 cookie=0x119, duration=60561.357s, table=1, n_packets=6, n_bytes=252, priority=152,arp,in_port=1,arp_tpa=192.168.29.55 actions=goto_table:2
 cookie=0xcc, duration=61107.999s, table=1, n_packets=7, n_bytes=294, priority=113,arp,in_port=1,arp_tpa=192.168.29.0/24 actions=goto_table:5
 cookie=0xd7, duration=61107.991s, table=1, n_packets=3, n_bytes=294, priority=116,ip,in_port=1,nw_dst=192.168.29.0/24 actions=goto_table:5
 cookie=0xf4, duration=61107.950s, table=1, n_packets=0, n_bytes=0, priority=143,arp,in_port=1,arp_tpa=192.168.122.0/24 actions=output:2
 cookie=0xf5, duration=61107.943s, table=1, n_packets=0, n_bytes=0, priority=146,ip,in_port=1,nw_dst=192.168.122.0/24 actions=output:2
 cookie=0xde, duration=61107.891s, table=1, n_packets=0, n_bytes=0, priority=112,tcp,in_port=1,nw_dst=192.168.29.176,tp_dst=22 actions=drop
 cookie=0xdf, duration=61107.884s, table=1, n_packets=0, n_bytes=0, priority=113,tcp,in_port=1,nw_dst=192.168.112.99,tp_dst=80 actions=drop
 cookie=0x111, duration=61107.957s, table=1, n_packets=5397, n_bytes=1121566, priority=21 actions=goto_table:2
```

3. VNF Creation QoS

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep table=2,
 cookie=0x174, duration=61526.234s, table=2, n_packets=0, n_bytes=0, priority=251,ip,in_port=1,nw_src=192.168.29.55 actions=set_queue:3,output:1
 cookie=0x179, duration=61526.226s, table=2, n_packets=0, n_bytes=0, priority=200,ip,in_port=1,nw_src=192.168.29.68 actions=set_queue:2,output:2
 cookie=0x17a, duration=61526.218s, table=2, n_packets=0, n_bytes=0, priority=200,ip,in_port=1,nw_src=192.168.122.38 actions=set_queue:2,output:2
 cookie=0x17e, duration=60958.910s, table=2, n_packets=0, n_bytes=0, priority=251,arp,in_port=1,arp_spa=192.168.29.55 actions=set_queue:3,output:1
 cookie=0x173, duration=61526.241s, table=2, n_packets=0, n_bytes=0, priority=250,ip,in_port=1,nw_dst=192.168.29.55 actions=set_queue:3,output:2
 cookie=0x17d, duration=60945.626s, table=2, n_packets=6, n_bytes=252, priority=250,arp,in_port=1,arp_tpa=192.168.29.55 actions=set_queue:3,output:2
/ #
```

Level 3: VNF Creation OVS Switch - Docker

4. VNF Creation VLAN

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep table=5,
 cookie=0x21d, duration=61628.199s, table=5, n_packets=10, n_bytes=588, priority=501,in_port=1,vlan_tci=0x0000/0x1fff actions=push_vlan:0x8100,set_field:4116->vlan_vid,outp
ut:2
 cookie=0x21f, duration=61627.251s, table=5, n_packets=7, n_bytes=546, priority=503,in_port=2,dl_vlan=20 actions=pop_vlan,output:1
```

5. VNF Creation L2

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data
 cookie=0xcc, duration=86029.594s, table=0, n_packets=114, n_bytes=6580, priority=103,in_port=eth1 actions=output:eth3
 cookie=0xce, duration=86028.968s, table=0, n_packets=45, n_bytes=3542, priority=105,in_port=eth3 actions=output:eth1
```

6. VNF Creation ARP

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep arp
 cookie=0xc8, duration=17.905s, table=0, n_packets=0, n_bytes=0, priority=200,arp,in_port=2 actions=CONTROLLER:6653
```

7. VNF Creation DNS

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep udp
 cookie=0xc9, duration=62.639s, table=0, n_packets=0, n_bytes=0, priority=201,udp,in_port=2,tp_dst=53 actions=CONTROLLER:6653
/ #
```

Level 3: Performance Results- Docker

1. Intra-domain reachability

```
[root@host1 ~]# ping -I 192.168.29.68 192.168.29.176
PING 192.168.29.176 (192.168.29.176) from 192.168.29.68 : 56(84) bytes of data.
64 bytes from 192.168.29.176: icmp_seq=1 ttl=64 time=12.6 ms
64 bytes from 192.168.29.176: icmp_seq=2 ttl=64 time=1.45 ms
64 bytes from 192.168.29.176: icmp_seq=3 ttl=64 time=1.17 ms
64 bytes from 192.168.29.176: icmp_seq=4 ttl=64 time=0.961 ms
64 bytes from 192.168.29.176: icmp_seq=5 ttl=64 time=1.17 ms
64 bytes from 192.168.29.176: icmp_seq=6 ttl=64 time=23.4 ms
64 bytes from 192.168.29.176: icmp_seq=7 ttl=64 time=1.14 ms
64 bytes from 192.168.29.176: icmp_seq=8 ttl=64 time=1.19 ms
64 bytes from 192.168.29.176: icmp_seq=9 ttl=64 time=0.953 ms
64 bytes from 192.168.29.176: icmp_seq=10 ttl=64 time=3.84 ms
64 bytes from 192.168.29.176: icmp_seq=11 ttl=64 time=1.13 ms
^C
--- 192.168.29.176 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10016ms
rtt min/avg/max/mdev = 0.953/4.468/23.468/6.853 ms
```

2. SSH blocked by firewall

```
[root@host1 ~]# ssh root@192.168.29.176 -I 192.168.29.68
ssh: connect to host 192.168.29.176 port 22: Connection timed out
```

```
[root@host1 ~]# ssh root@192.168.29.176
ssh: connect to host 192.168.29.176 port 22: Invalid argument
[root@host1 ~]#
```

3. Reachability blocked by firewall rule

```
[root@host1 ~]# ping -I 192.168.29.68 192.168.29.99
PING 192.168.29.99 (192.168.29.99) from 192.168.29.68 : 56(84) bytes of data.
From 192.168.29.68 icmp_seq=1 Destination Host Unreachable
From 192.168.29.68 icmp_seq=2 Destination Host Unreachable
From 192.168.29.68 icmp_seq=3 Destination Host Unreachable
From 192.168.29.68 icmp_seq=4 Destination Host Unreachable
From 192.168.29.68 icmp_seq=5 Destination Host Unreachable
From 192.168.29.68 icmp_seq=6 Destination Host Unreachable
From 192.168.29.68 icmp_seq=7 Destination Host Unreachable
From 192.168.29.68 icmp_seq=8 Destination Host Unreachable
^C
--- 192.168.29.99 ping statistics ---
10 packets transmitted, 0 received, +8 errors, 100% packet loss, time 9001ms
```

4. VLAN creation

```
/ # ovs-ofctl dump-flows br-data -O OpenFlow13 | grep table=5
 cookie=0x21d, duration=59.101s, table=5, n_packets=20, n_bytes=1728, priority=501,in_port=1,vlan_tci=0x0000/0x1fff actions=push_vla
:2
 cookie=0x21f, duration=58.428s, table=5, n_packets=18, n_bytes=1626, priority=503,in_port=2,dl_vlan=20 actions=pop_vlan,output:1
/ #
```