Spring Final 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

# PROJECT PURPOSE & OBJECTIVE

# Project Objectives

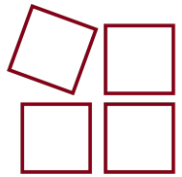
## Objective

- 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

# Project Purpose



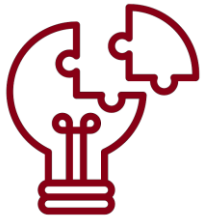
## Field of Application

- ✓ **Enterprise Network:** Smart employee devices which can provide complex network services on the host devices
- ✓ **Datacenter:** Smart servers which will now provide networking capability
- ✓ **CDN:** Cache servers with sophisticated networking functionality
- ✓ **5G:** Cheaper implementation of network elements with NFV and white boxes.
- ✓ **IoT Device Network:** Cheaper and faster networking for IoT devices using containerized VNFs

## Potential Impact

- ✓ Novel idea of bringing complex network functions to be deployed using lightweight NFVs
- ✓ Removes the single point of failure of network functions
- ✓ Granularity in user experience
- ✓ One-touch orchestration/ZTP of network services on hosts

# Project Purpose (continued…)



## Problem Addressed

- ✓ **Cost:** Proprietary network gears are very expensive
- ✓ **Complexity:** Rid of complex networks design
- ✓ **Usage:** Efficient resource usage of smart hosts and servers
- ✓ **CI/CD:** Faster service deployment and testing
- ✓ **VAS:** Ease in providing customized value-added services
- ✓ **QoS:** Reduce latency and increased throughput
- ✓ **User Experience:** Enhanced user experience
- ✓ **Deployment Scope:** Lightweight device/OS independent setup

# Levels of Success

Deployment Testing

Integration Testing

Compatibility Testing

## Configuration

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

Performance Testing

## Performance

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

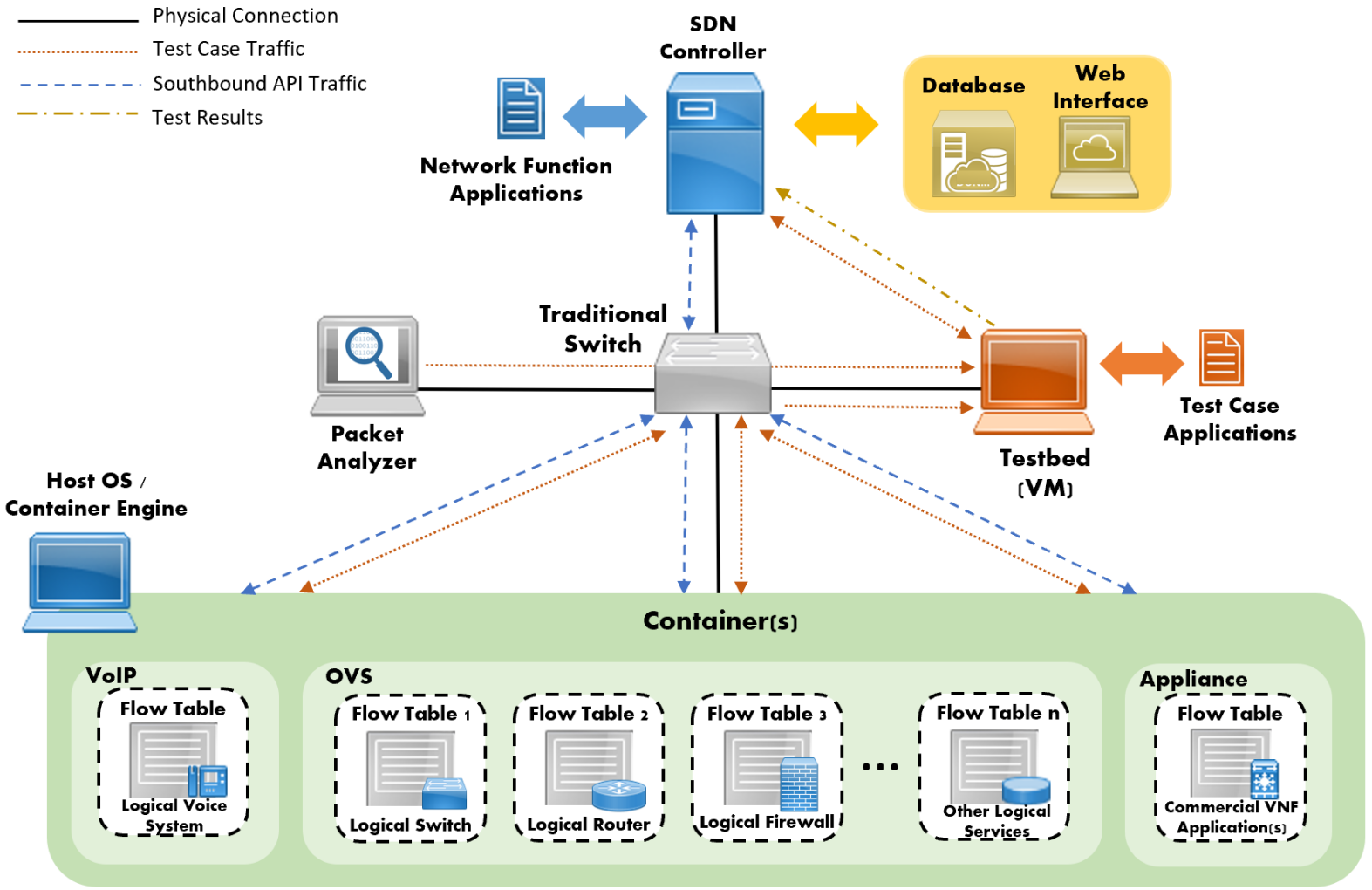
Stress Testing

Multi Environment Testing

## Failover

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

# Concept of Operations (CONOPS)



# DESIGN DESCRIPTION

# Critical Project Elements - Technical

**CPE 1.1: VNF Creation**

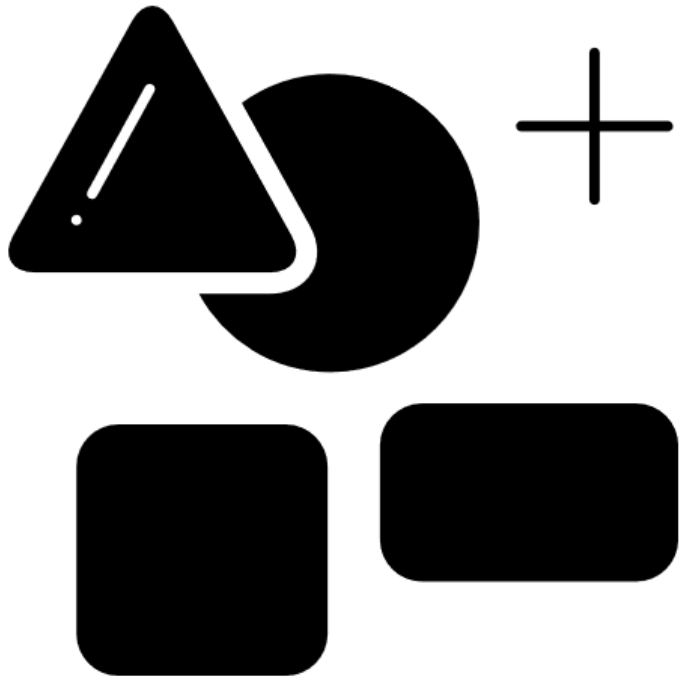
- Deploy OVS Switch using containers
- Create VNFs corresponding to network services on OVS Switches

**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



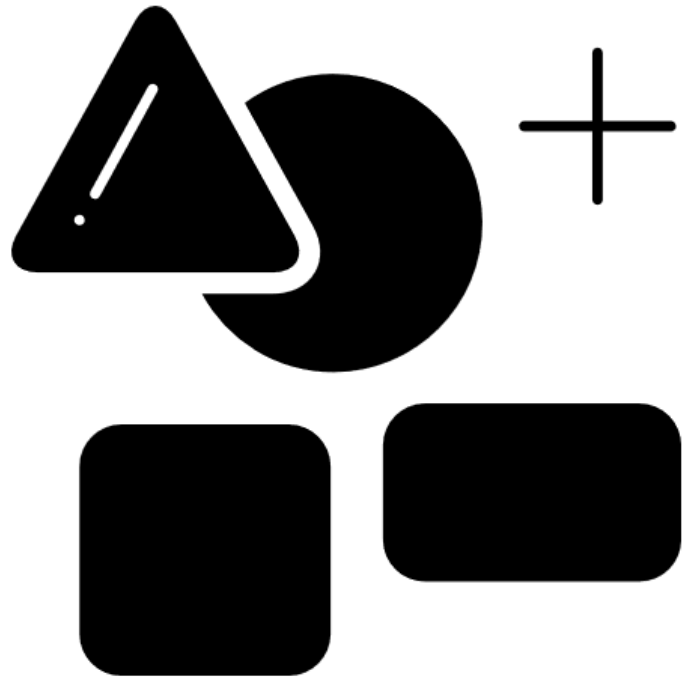
# Critical Project Elements – Technical (continued)

**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**

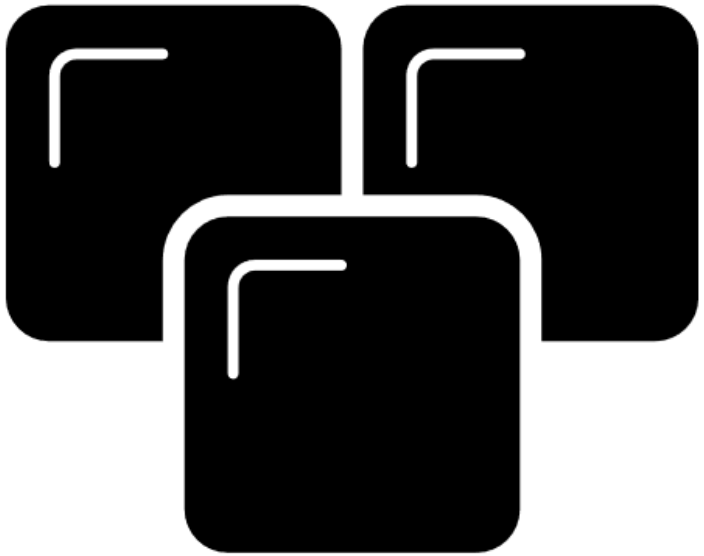
- Using servers and traditional switches available in Telecommunications lab
- Laptops with Windows, Linux and MacOS as host devices

**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**

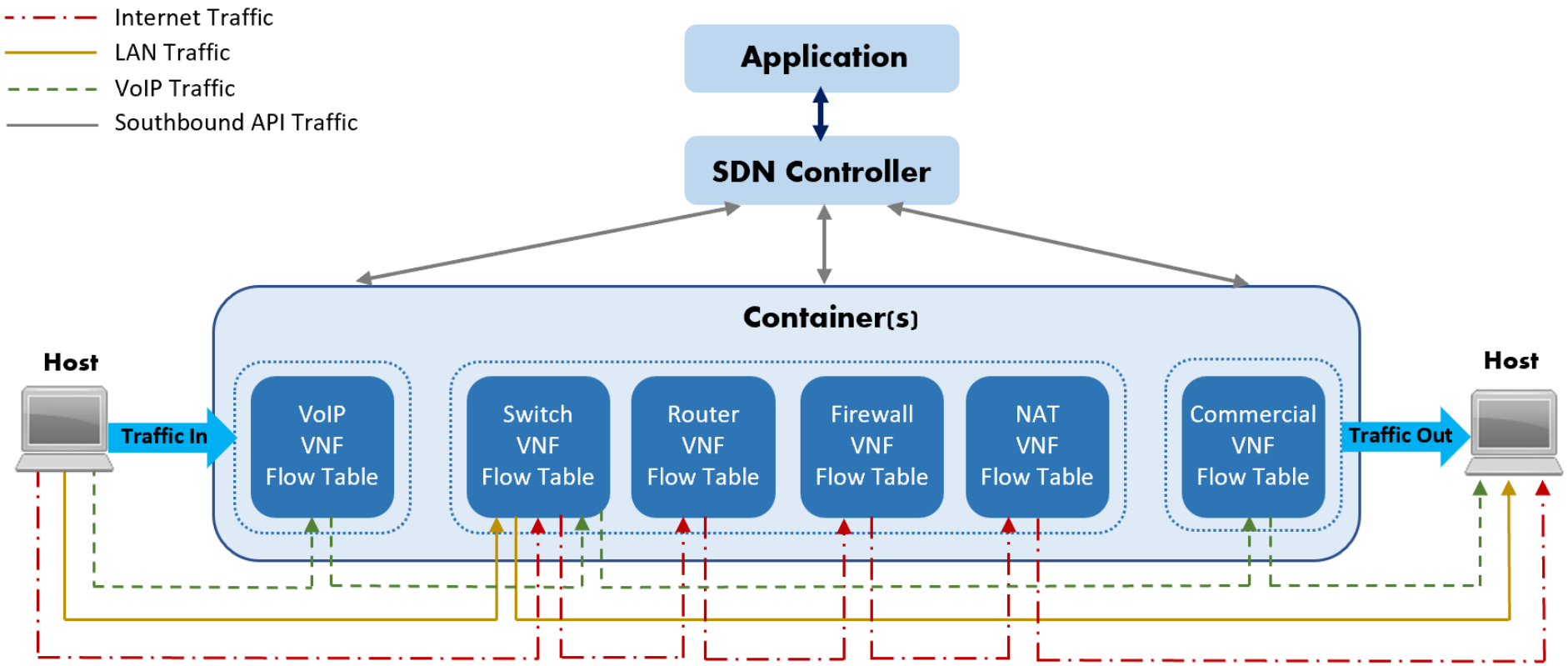
- Understand the functionality of containers
- Determine web GUI framework and learn the functionality



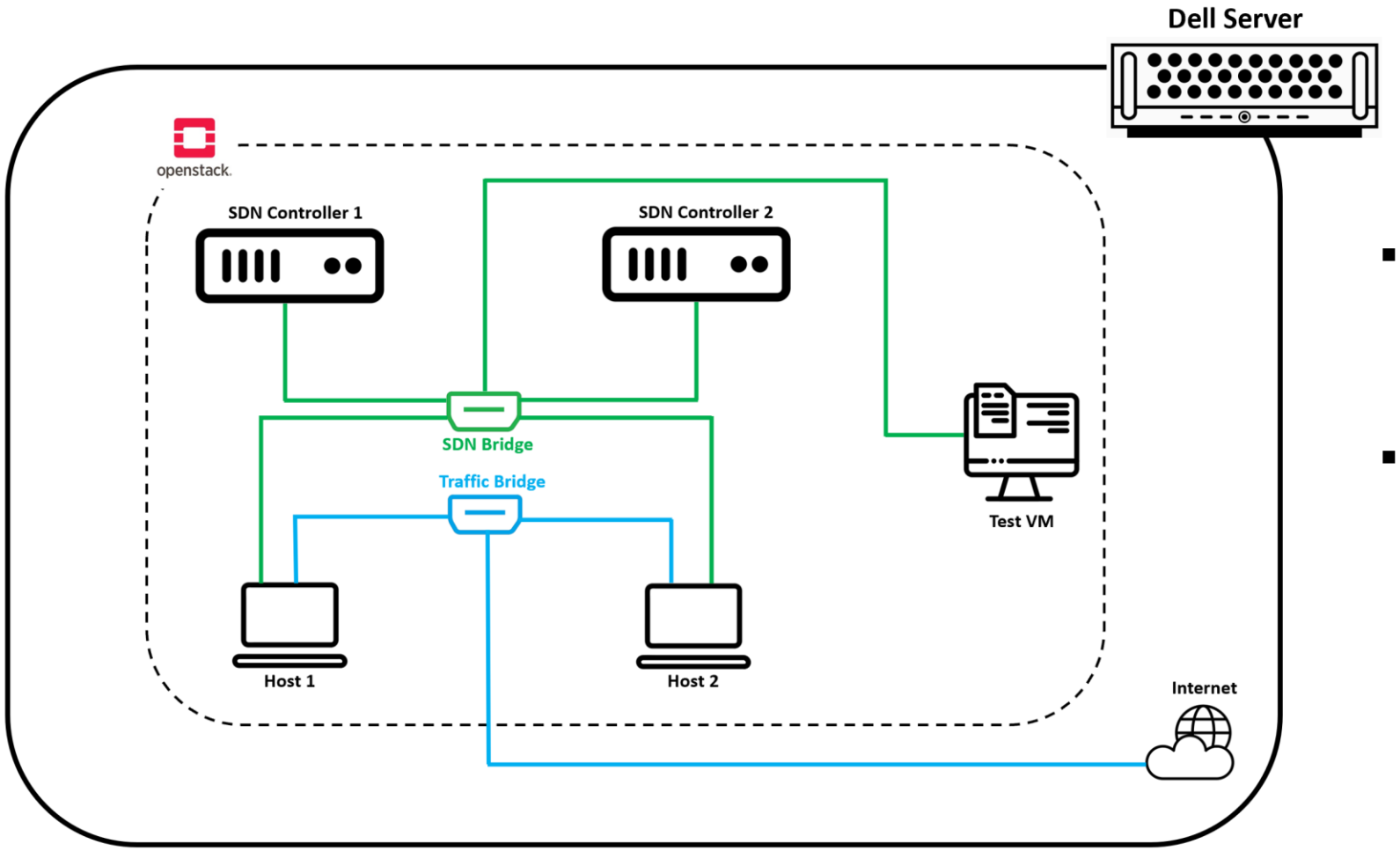
# Design Description

| Design Goals | Design Solution |
| --- | --- |
| Virtualization of Network Services to reduce intermediate device overhead. | Install flow entries in flow-tables on OvS switches where each flow table represents a unique network function. |
| Reduce Time to Deploy and enhance network performance | Programmable network functions, Orchestration and Abstraction |

# Functional Block Diagram (FBD)

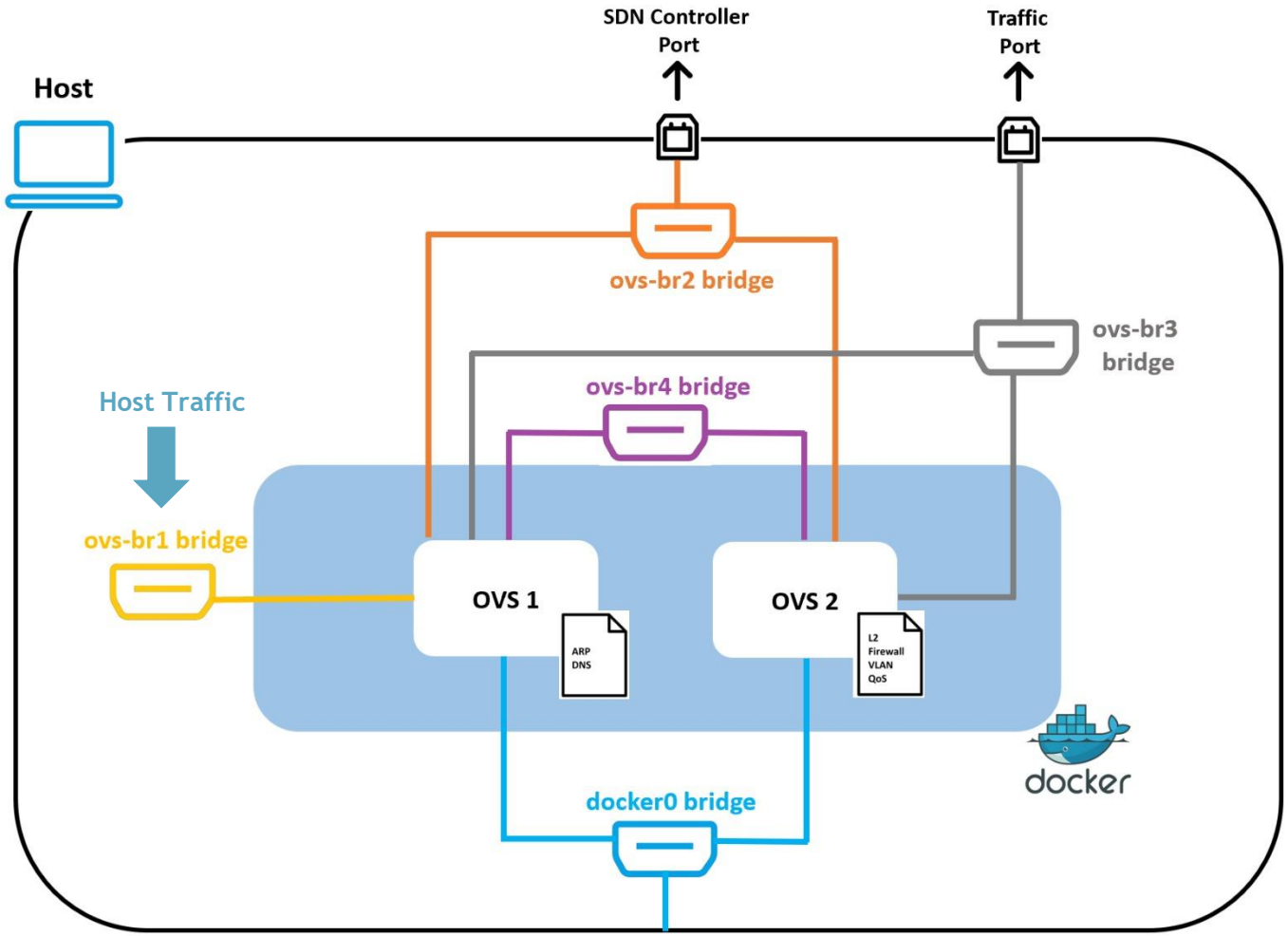


# Project Implementation - Network




- 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*

# Project Components

| Type | Product | Features | Description |
|---|---|---|---|
| Hardware | Dell PowerEdge R430 | Memory: 64 GB<br>Networking: 4X1Gb Ethernet NICs<br>CPU: Intel® Xeon® E5-2600 v4 family | Telecom Lab |
| Software | Ubuntu | Version: 16.04.6 | Operating System |
| | OpenStack | Version: Stable/Rocky | Cloud Orchestrator |
| | OvS | Programmable Switch | OpenFlow enabled MLS |
| | Docker Swarm | Version: 19.03.7 | Container Orchestrator |
| VNF | Layer2 | Forwarding | Intra-NW-Domain Reachability |
| | Layer3 | Routing | Inter-NW-Domain Reachability |
| | NAT | IP Translation | Internet Reachability |
| | ARP | MAC Address Resolution | Layer 2 Address Mapping |
| | FIREWALL | Restrict Traffic | Network Security |
| | VLAN | Traffic Segregation | Differentiate Internal and Outbound Traffic |
| | QoS | Traffic Precedence | Prioritize different Traffic Types |

# TEST OVERVIEW

# Test Overview Outline

| | | | | |
|---|---|---|---|---|
| 1 | DEPLOYMENT TEST | Test configuration and deployment of Network Devices and Software's. | Manual/Automated | Success |
| 2 | INTEGRATION TEST | Test and Verify network deployment and device reachability | Manual/Automated | Success |
| 3 | COMPATIBILITY TEST | Test and verify the operation of network protocol stack | Manual/Automated | Success |
| 4 | PERFORMANCE TEST | Test, verify and validate VNF functionality and service chaining. | Manual/Automated | Success |
| 5 | STRESS TEST | Test to verify fail-over mechanism of network entities | Manual/Automated | Success |
| 6 | MULTI-ENVIRONMENT TEST | Verify operability of the solution on multiple Operating system | Manual | Success |

# DEPLOYMENT TEST OVERVIEW & RESULTS

# Test Overview – Deployment Test

- **Test Case Objective:**
  - To validate the consistency of underlying network infrastructure and tools
- **Test Case Result:**
  - Verify and validate the successful deployment of network infrastructure and tools such as OpenStack, VM, SDN Controller, Open vSwitch which supports the VNFs and service chain operability
- **Test Case Significance:**
  - The results verify the availability of these crucial network elements. Builds the platform for future tests

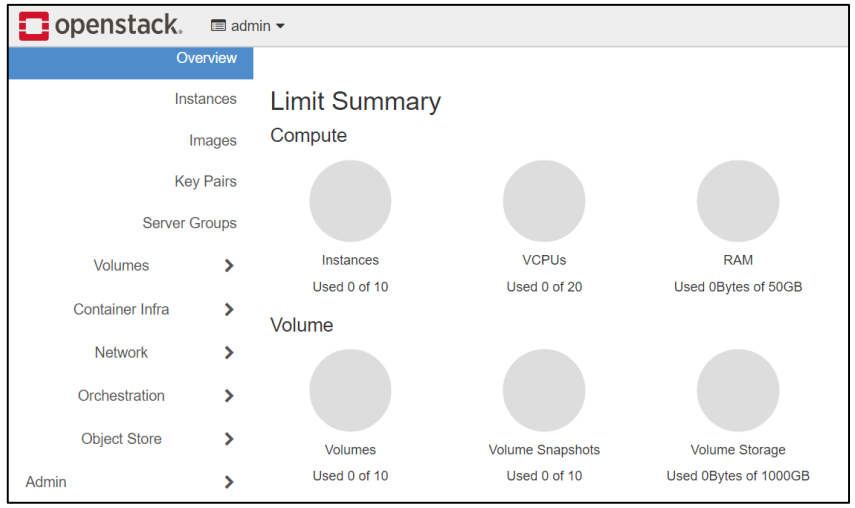
# Test Cases and Strategy – Deployment Test

| Test Case | Strategy | Expected Output | Result |
|---|---|---|---|
| Openstack Deployment | Verify Dashboard accessibility and OpenStack version | “OpenStack -V” command will return the version for successful installation | PASS |
| Virtual Machine Configuration | Verify via Command-Line, Openstack Dashboard and VM reachability | “openstack server list” shows the list of VMs along with assigned IP’s. Ping is used to ensure connectivity | PASS |
| Network Software Installation | Verify the installation of required network software by checking the version | Successfully installation would display version information via CLI. Ex: “docker –V” | PASS |
| OpenvSwitch Installation | Verify Container status | “docker ps –a” and “docker inspect” displays OvS setup and configuration status | PASS |
| SDN Controller installation | Verify containerization of SDN controller | “docker ps –a” and “docker images” displays information of the installed Controller | PASS |

# Level 1: Deployment of Network Infrastructure

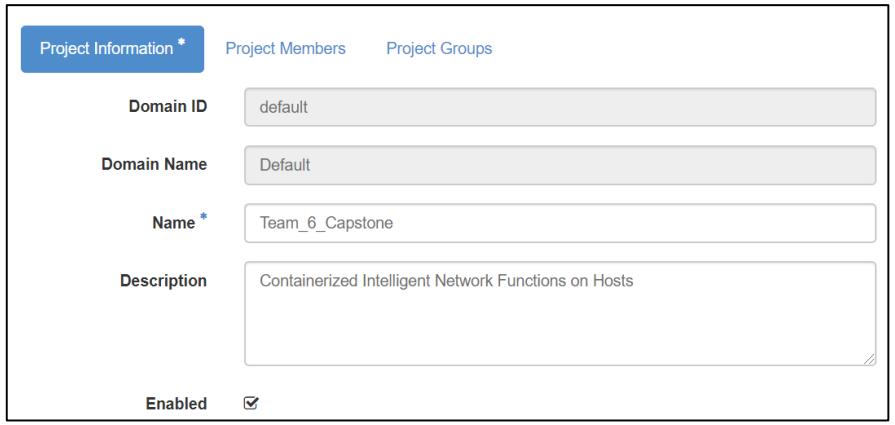
- **Deployment of OpenStack on the Dell Server**

1. OpenStack GUI Dashboard

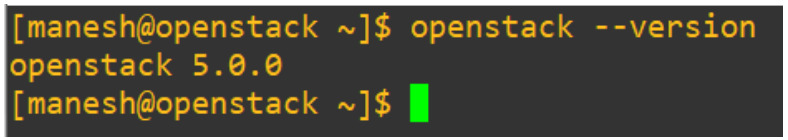


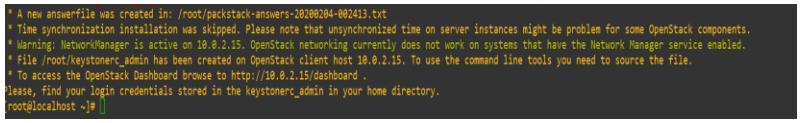

2. OpenStack Setup Logs


3. OpenStack Project Details




4. Project Details CLI



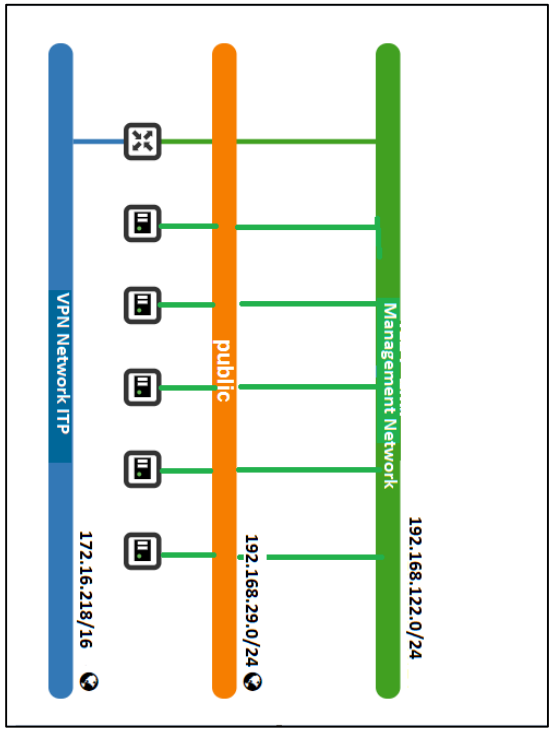
# Level 1: Deployment of Network Infrastructure

- **Instance Creation and Address Allocation**

## 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     |
+--------------------+----------------------+--------+------------+-------------+-------------------+--------------------+
```

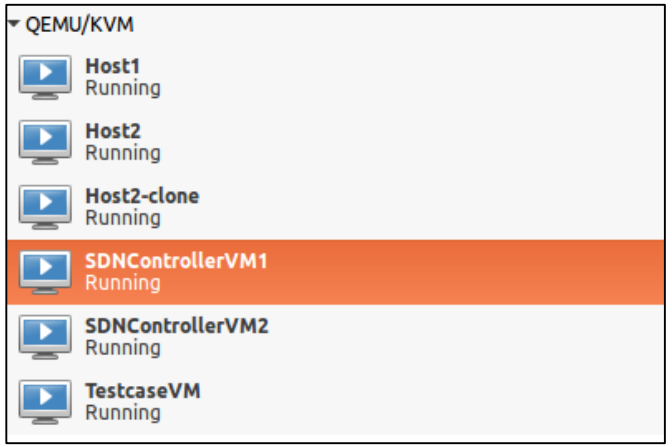
## 2. OpenStack Dashboard View




## 3. 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
```

## 4. Hypervisor Topology View



# Level 1: Deployment of Network Infrastructure

- **Network Tools Deployment**

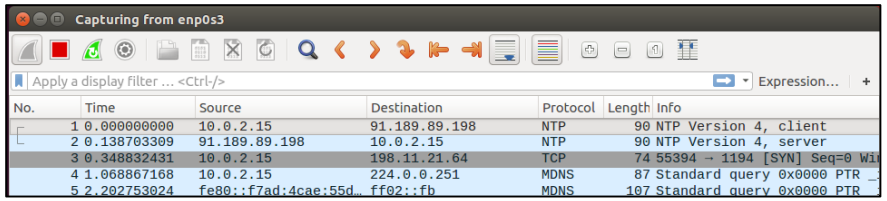
## 1. Docker installation

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


## 2. Wireshark installation

```
manesh@manesh-VirtualBox:~$ wireshark --version
Wireshark 2.6.10 (Git v2.6.10 packaged as 2.6.10-1~ubuntu16.04.0)

Copyright 1998-2019 Gerald Combs <gerald@wireshark.org> and contributors.
License GPLv2+: GNU GPL version 2 or later <http://www.gnu.org/licenses/old-
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```




## 3. Git installation

```
manesh@manesh-VirtualBox:~$ git --version
git version 2.7.4
manesh@manesh-VirtualBox:~$
```

```
manesh@manesh-VirtualBox:~$ git config --list
user.name=maya8094
user.email=maya8094@colorado.edu
manesh@manesh-VirtualBox:~$
```

```
manesh@manesh-VirtualBox:~$ git remote add origin_team6 https://github.com/opennetwork
manesh@manesh-VirtualBox:~$ git pull origin_team6
warning: no common commits
remote: Enumerating objects: 35423, done.
remote: Total 35423 (delta 0), reused 0 (delta 0), pack-reused 35423
Receiving objects: 100% (35423/35423), 177.50 MiB | 26.29 MiB/s, done.
Resolving deltas: 100% (20700/20700), done.
From https://github.com/opennetworkinglab/OpenNetworkLinux
 * [new branch]      EC-ONLPv2-wedgebf-32x -> origin_team6/EC-ONLPv2-wedgebf-32x
 * [new branch]      ONLPv2      -> origin_team6/ONLPv2
 * [new branch]      ONLPv2-bazel -> origin_team6/ONLPv2-bazel
 * [new branch]      d10         -> origin_team6/d10
 * [new branch]      d5254-kernel-upgrade -> origin_team6/d5254-kernel-upgrade
 * [new branch]      docker-support -> origin_team6/docker-support
 * [new branch]      fix-692     -> origin_team6/fix-692
 * [new branch]      gh-pages    -> origin_team6/gh-pages
 * [new branch]      master      -> origin_team6/master
 * [new branch]      onf-ONLPv2 -> origin_team6/onf-ONLPv2
 * [new branch]      systemd     -> origin_team6/systemd
 * [new tag]         onlpv2-dev-1.1.0 -> onlpv2-dev-1.1.0
 * [new tag]         onlpv2-dev-1.1.1 -> onlpv2-dev-1.1.1
You asked to pull from the remote 'origin_team6', but did not specify
a branch. Because this is not the default configured remote
for your current branch, you must specify a branch on the command line.
manesh@manesh-VirtualBox:~$
```

## 3. iPerf installation

```
manesh@manesh-VirtualBox:~$ iperf --version
iperf version 2.0.5 (2 June 2018) pthreads
manesh@manesh-VirtualBox:~$
```

## 4. Python installation

```
manesh@manesh-VirtualBox:~$ python --version
Python 2.7.12
manesh@manesh-VirtualBox:~$
```

# Level 1: Deployment of Network Infrastructure

- **Docker-based Software Deployment**

## 1. SDN Controller Installation

```
[root@localhost ~]# docker run -it --rm --name sdn-ryu osrg/ryu /bin/bash
Unable to find image 'osrg/ryu:latest' locally
Trying to pull repository docker.io/osrg/ryu ...
latest: Pulling from docker.io/osrg/ryu
34667c7e4631: Pull complete
d18d76a881a4: Pull complete
119c7358fbfc: Pull complete
2aaf13f3eff0: Pull complete
d9a4985f3aca: Pull complete
Digest: sha256:863c63d95a53f45ebdc7b6f111af0032af275e427e0d8d26355f6fc8e57747c8
Status: Downloaded newer image for docker.io/osrg/ryu:latest
root@b6fff6c266ba:~#
```

```
[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]#
```

## 2. Open vSwitch installation

```
[root@localhost ~]# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
docker.io/globocom/openvswitch      latest              58215e48af7c        2 years ago         73.4 MB
[root@localhost ~]#
```

```
[root@localhost ~]# docker run -it --rm --name vswitch --cap-add=NET_ADMIN -d globocom/openvswitch
4a8520d6fd08c8452bfa1f8391b4ba4ca904e9e340794c5f4daae3af750c4759
[root@localhost ~]#
```

```
[root@localhost ~]# docker run -it --rm --name vswitch1 --cap-add=NET_ADMIN -d globocom/openvswitch
bb88a33de9844a2180c5c165a94bd82a722a5e6e6e3fc744c180d599b60cde38
[root@localhost ~]#
```

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                   COMMAND                  CREATED              STATUS              PORTS               NAMES
4a8520d6fd08        globocom/openvswitch    "/bin/sh -c /usr/b..."   About a minute ago   Up About a minute   22/tcp              vswitch
bb88a33de984        globocom/openvswitch    "/bin/sh -c /usr/b..."   3 minutes ago        Up 3 minutes        22/tcp              vswitch1
[root@localhost ~]#
```

## 3. VNF Creation – Open vSwitch

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data
 cookie=0x6e, duration=7.468s, table=0, n_packets=0, n_bytes=0, priority=20,in_port=1 actions=output:2
 cookie=0x6f, duration=6.446s, table=0, n_packets=0, n_bytes=0, priority=20,in_port=2 actions=output:1
 cookie=0x0, duration=60.182s, table=0, n_packets=0, n_bytes=0, priority=0 actions=NORMAL
/ #
```

# PERFORMANCE TEST OVERVIEW & RESULTS

# Test Overview – Performance Test

- **Test Case Objective:**
  - To validate the service chain combinations
- **Test Case Results:**
  - Verify the performance and operability of network functions such as QoS, routing, firewall, VLANs, and NAT
  - Validate the functionality of the VNFs and the seamless performance of the service chains
- **Test Case Implication:**
  - The successful results sets base work towards the implementation on industry scale to achieve optimum network performance

# Test Cases and Strategy – Performance Test

| Test Case | Strategy | Expected Output | Result |
|---|---|---|---|
| Address Resolution Protocol | Verify the dynamic mapping of MAC addresses and IP addresses and their Static assignment on hosts | “arp -a” displays IP to MAC mapping for both Dynamic and Static assignment | PASS |
| Firewall | Firewall based flow entries to verify Complete Block, Partial Block and No Block data traffic.<br>1. Complete Block – Traffic destined to or originating from a particular IP is blocked. Eg: Ping & SSH<br>2. Partial Block – Certain traffic requests for a particular service is allowed. Eg: Ping<br>3. No Block– Traffic destined to or originating from a particular IP is allowed to access all services. Eg: Ping, SSH, HTTP | 1. Complete Block – SSH, ping and HTTP connectivity to the host address will fail.<br>2. Partial Block – SSH and HTTP connectivity to the host address will fail but ping connectivity will be successful.<br>3. No Block – SSH, ping and HTTP connectivity to the host address will be successful. | PASS |

# Test Cases and Strategy – Performance Test

| Test Case | Strategy | Expected Output | Result |
|---|---|---|---|
| Quality of Service | Verify priority treatment for certain traffic types destined to multiple network services through Wireshark capture. | Wireshark capture displays diffserv column in IP Header which contains the QoS class information. High priority traffic will have higher queue number. Non-queue based traffic from hosts will have default queue assigned | PASS |
| Virtual LAN | Verify successful traffic segregation through Vlan tagging in Wireshark capture | Wireshark capture displays VLAN tagging information for tagged traffic and no information for untagged | PASS |
| DNS query | Verify the authenticity of URL address requested (malicious or genuine) | 1. **Malicious URL:** Redirects the request to local web page displaying URL block message<br>2. **Genuine URL:** redirects the request to the original web page | PASS |

# Level 4: Performance Test for VNFs

- **Performance Test – ARP Setup and Functionality**

1. Dynamic ARP Mapping

- SDN Controller periodically maps the IP address to the MAC address of the device interfaces in the topology
- The mapping is periodically pushed into the ARP table of each host as static entries
- This mapping saves the time in ARP related queries and reduces latency
- For an unknown IP Address with no ARP entry in the ARP table, query is sent to the SDN Controller

2. VNF Creation – ARP (Flow Table 0 – Vswitch1)

```
/ # 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
```

3. ARP Mapping

```
{'192.168.122.244': '82:5a:01:22:46:8b',
 '192.168.122.142': '72:8a:b0:8d:fd:ec',
 '192.168.122.117': '1a:4e:70:41:c1:86',
 '192.168.122.38': '5a:bc:ae:1a:e2:a7'}
```

4. Static ARP Entries - ARP Table

| Address | HWtype | HWaddress |
|---|---|---|
| 192.168.122.142 | ether | 72:8a:b0:8d:fd:ec |
| 192.168.122.244 | ether | 82:5a:01:22:46:8b |
| 192.168.122.117 | ether | 1a:4e:70:41:c1:86 |
| 192.168.122.38 | ether | 5a:bc:ae:1a:e2:a7 |

# Level 4: Performance Test for VNFs

- **Performance Test – DNS Setup and Functionality**

1. VNF Creation - DNS (Flow Table 0 – Vswitch1)

- DNS related query are sent to the SDN Controller
- **URL's in the white-list:** Corresponding IP Address is sent in the Packet_Out message
- The HTTP requests are forwarded to the other VNFs in the service chain
- **URL's in the black-list:** IP Address of a local web page is sent in the Packet_Out message
- The HTTP requests for the blacklist URL's are forwarded to the local web page
- The local web page displays the URL block message for the user to know

```
/ # 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
/ #
```

2. Reactive flows added for white-list and black-list URL's

```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=9.533s, table=0, n_packets=0, n_bytes=0, idle_timeout=10, idle_age=9, priority=4100
0,in_port=3,dl_dst=ff:ff:ff:ff:ff:ff actions=mod_dl_dst:00:15:f9:0c:85:71,mod_nw_dst:172.16.0.1,output:2
 cookie=0x0, duration=9.530s, table=0, n_packets=15, n_bytes=1252, idle_timeout=10, idle_age=0, priority=
40000,in_port=3 actions=output:2,CONTROLLER:65509
 cookie=0x0, duration=9.526s, table=0, n_packets=8, n_bytes=1261, idle_timeout=10, idle_age=1, priority=4
0000,in_port=2 actions=output:3,CONTROLLER:65509
```

# Level 4: Performance Test for VNFs

- **Performance Test – DNS Setup and Functionality**

1. White-list URL forwarding

```
NEW REQUEST: The identified domain is ===== goodweb.herokuapp.com
This is the checker url http://checksite.herokuapp.com/api/url=http://goodweb.herokuapp.com
This is the http response: good
---------- Good website requested ----------
-------- Good website resolved and served ---------
```

```
> User Datagram Protocol, Src Port: 53, Dst Port: 60793
v Domain Name System (response)
    Transaction ID: 0xc58a
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 0
  v Queries
    > goodweb.herokuapp.com: type A, class IN
  v Answers
    > goodweb.herokuapp.com: type A, class IN, addr 52.3.97.93
  > Authoritative nameservers
    [Request In: 11]
    [Time: 0.141947000 seconds]
```

2. Black-list URL blocking

```
NEW REQUEST: The identified domain is ===== badweb.herokuapp.com
This is the checker url http://checksite.herokuapp.com/api/url=http://badweb.herokuapp.com
This is the http response: bad
---------- Bad website requested ----------
122194266805
{'priority': 40000, 'actions': [{'type': 'DROP'}], 'match': {'nw_proto': 6, 'nw_dst': '34.23
dpid': 122194266805}
(21555) accepted ('10.20.30.44', 58434)
Unknown action type
10.20.30.44 - - [09/Dec/2019 19:47:35] "POST /stats/flowentry/add HTTP/1.1" 200 139 0.002143
Status code of flow install: 200
---------Flow for bad DNS request installed---------
Redirected to local web server for bad website DNS request
```

# Level 4: Performance Test for VNFs

- **Performance Test – VLAN Setup and Functionality**

1. VNF Creation – VLAN (Flow Table 5 – Vswitch1)

- VLAN assignment is used to segregate the traffic according to the hosts
- Intra-domain traffic for network 192.168.29.0/24 is segregated with VLAN20
- Rest of the traffic remains untagged
- Connectivity is ensured for both VLAN-tagged and untagged traffic
- Traffic requests are forwarded to VLAN VNF (flow table 5) post firewall VNF clearance

```
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=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
```

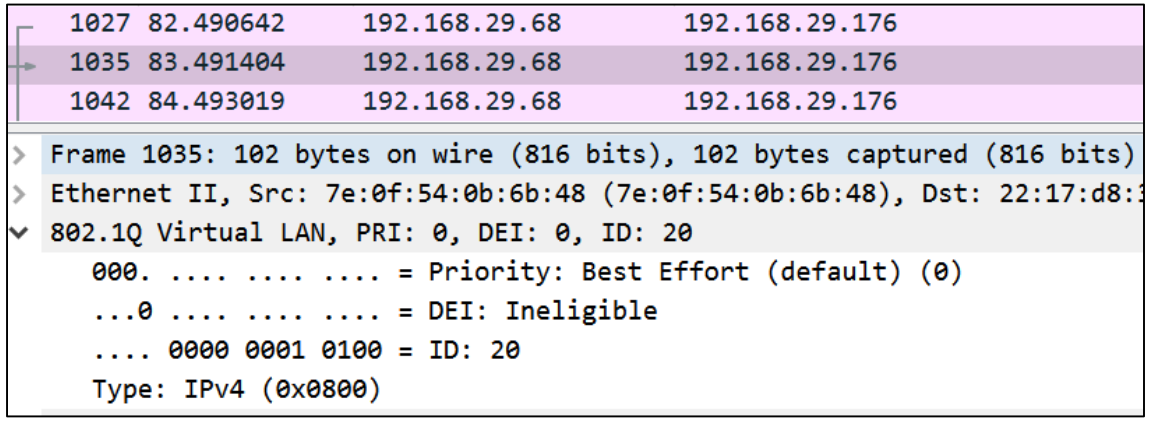
```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep table=5,
 cookie=0x21d, duration=61628.199s, table=5, n_packets=10, n_bytes=588, priority=501,in_port=1,vlan_tci=0x0000/0x1fff actions=push_vlan:0x8100,set_field:4116->vlan_vid,outp
ut:2
 cookie=0x21f, duration=61627.251s, table=5, n_packets=7,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

# Level 4: Performance Test for VNFs

## 1. Tagged VLAN Connectivity

```
[khir@host1 ~]$
[khir@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=5.46 ms
64 bytes from 192.168.29.176: icmp_seq=2 ttl=64 time=1.47 ms
64 bytes from 192.168.29.176: icmp_seq=3 ttl=64 time=1.06 ms
^C
--- 192.168.29.176 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.065/2.668/5.461/1.982 ms
[khir@host1 ~]$
```

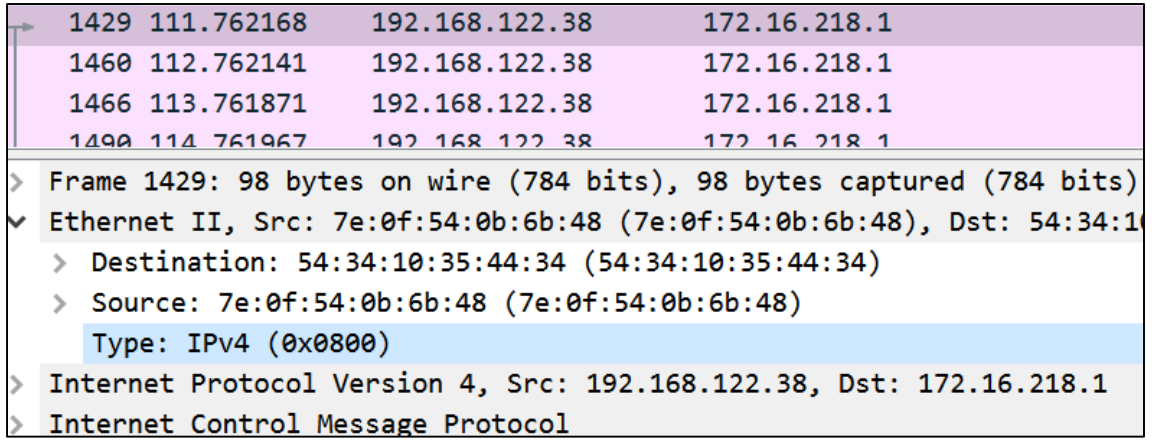
## 2. Wireshark Capture – Tagged VLAN




## 3. Untagged VLAN Connectivity

```
[khir@host1 ~]$
[khir@host1 ~]$ ping 172.16.218.1
PING 172.16.218.1 (172.16.218.1) 56(84) bytes of data.
64 bytes from 172.16.218.1: icmp_seq=1 ttl=64 time=2.03 ms
64 bytes from 172.16.218.1: icmp_seq=2 ttl=64 time=0.509 ms
64 bytes from 172.16.218.1: icmp_seq=3 ttl=64 time=0.457 ms
64 bytes from 172.16.218.1: icmp_seq=4 ttl=64 time=0.671 ms
^C
--- 172.16.218.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.457/0.918/2.037/0.651 ms
[khir@host1 ~]$
```

## 2. Wireshark Capture – Untagged VLAN



# Level 4: Performance Test for VNFs

- **Performance Test – QoS Setup and Functionality**

1. QoS Class Implementation

- 2 queue classes are defined for ingress interface
- 2 queue classes are defined for egress interface

```
/ # ovs-vsctl list qos
_uuid               : 875608b8-0405-436b-a821-0eaa4ac650cd
external_ids        : {}
other_config        : {}
queues              : {2=b3a2dfe4-44c4-43d1-8308-69e14f78757d, 3=e5d9dedc-7cf1-4fce-9f5b-e8fe074f7c3e}
type                : PRONTO_ROUND_ROBIN

_uuid               : fe833600-f0fb-45bb-91ea-fddf5966d157
external_ids        : {}
other_config        : {}
queues              : {2=160d5b94-70c1-43d0-b875-5c52d38e8acf, 3=6f72ee78-c0ba-448d-b0d0-f1194a7599cd}
type                : PRONTO_ROUND_ROBIN
```

```
name                : "eth2"
ofport              : 2
statistics          : {collisions=0, rx_bytes=2303996597, rx_crc_err=0, rx_dropp
ed=315240, rx_errors=0, rx_frame_err=0, rx_over_err=0, rx_packets=22324088, tx_b
ytes=2835362272, tx_dropped=12760, tx_errors=0, tx_packets=35520403}
status              : {driver_name=veth, driver_version="1.0", firmware_version=
name                : "eth1"
ofport              : 1
statistics          : {collisions=0, rx_bytes=2835812742, rx_crc_err=0, rx_dropp
ed=28188, rx_errors=0, rx_frame_err=0, rx_over_err=0, rx_packets=35527613, tx_by
tes=2294938647, tx_dropped=129052, tx_errors=0, tx_packets=22234553}
status              : {driver_name=veth, driver_version="1.0", firmware_version=
""}
```

2. QoS Queues Implementation

```
/ # ovs-vsctl list queue b3a2dfe4-44c4-43d1-8308-69e14f78757d
_uuid               : b3a2dfe4-44c4-43d1-8308-69e14f78757d
dscp                : []
external_ids        : {}
other_config        : {max-rate="200000000", min-rate="100000000"}
/ #
```

```
/ # ovs-vsctl list queue e5d9dedc-7cf1-4fce-9f5b-e8fe074f7c3e
_uuid               : e5d9dedc-7cf1-4fce-9f5b-e8fe074f7c3e
dscp                : []
external_ids        : {}
other_config        : {max-rate="400000000", min-rate="150000000"}
/ #
```

```
/ # ovs-vsctl list queue 160d5b94-70c1-43d0-b875-5c52d38e8acf
_uuid               : 160d5b94-70c1-43d0-b875-5c52d38e8acf
dscp                : []
external_ids        : {}
other_config        : {max-rate="200000000", min-rate="100000000"}
/ #
```

```
/ # ovs-vsctl list queue 6f72ee78-c0ba-448d-b0d0-f1194a7599cd
_uuid               : 6f72ee78-c0ba-448d-b0d0-f1194a7599cd
dscp                : []
external_ids        : {}
other_config        : {max-rate="400000000", min-rate="150000000"}
/ #
```

# Level 4: Performance Test for VNFs

- **Performance Test – QoS Setup and Functionality**

1. VNF Creation – QoS (Flow Table 2 – Vswitch1)

- QoS classification is used to prioritize different service types for different hosts
- Hosts in queue 3 are assigned more bandwidth (200 MBPS)
- Hosts in queue 2 will get less bandwidth (100 MBPS)
- Rest of the packet requests will have default queue assigned

```
/ # 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
/ #
```

2. Wireshark Capture - Queue 2 Service Assignment

```
Internet Protocol Version 4, Src: 192.168.122.38, Dst: 192.168.
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not
    Total Length: 88
    Identification: 0xffe1 (65505)
```

3. Wireshark Capture – Default Queue Service Assignment

```
Internet Protocol Version 4, Src: 192.168.122.38, Dst:
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN:
    Total Length: 84
    Identification: 0xb322 (45858)
```

# Level 4: Performance Test for VNFs

- **Performance Test – Firewall Setup and Functionality**

1. VNF Creation – Firewall (Flow Table 1 – Vswitch1)

- Granular implementation of flows to allow or drop the traffic requests
- Complete service drop/block, Partial service drop/block and Complete service enable

```
/ # 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
```

# Level 4: Performance Test for VNFs

1. Complete Service Block

- All kind of traffic requests destined to or originated from a particular IP address are dropped/blocked
- Services include SSH/Ping/HTTP services

2. SSH Requests Dropped/Blocked

```
[khir@host1 ~]$ ssh root@4.4.4.4
ssh: connect to host 4.4.4.4 port 22: Connection timed out
[khir@host1 ~]$
```

3. Ping Requests Dropped/Blocked

```
[khir@host1 ~]$ ping 4.4.4.4
PING 4.4.4.4 (4.4.4.4) 56(84) bytes of data.
^C
--- 4.4.4.4 ping statistics ---
173 packets transmitted, 0 received, 100% packet loss, time 172008ms

[khir@host1 ~]$
```

# Level 4: Performance Test for VNFs

1. Partial Service Block

- A few kind of traffic requests destined to or originated from a particular IP address are dropped/blocked
- Other traffic requests are allowed without any issue
- Allowed services include Ping service
- Blocked services include SSH/HTTP services

2. SSH Services Dropped/Blocked

```
[khir@host1 ~]$
[khir@host1 ~]$ ssh root@192.168.29.176
ssh: connect to host 192.168.29.176 port 22: Connection timed out
```

3. Ping Request Allowed

```
[khir@host1 ~]$
[khir@host1 ~]$ ping 192.168.29.176
PING 192.168.29.176 (192.168.29.176) 56(84) bytes of data.
64 bytes from 192.168.29.176: icmp_seq=1 ttl=64 time=5.81 ms
64 bytes from 192.168.29.176: icmp_seq=2 ttl=64 time=1.60 ms
^C
--- 192.168.29.176 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.600/3.707/5.815/2.108 ms
[khir@host1 ~]$
```

# Level 4: Performance Test for VNFs

1. All Services Allowed

- All kind of traffic requests destined to or originated from a particular IP address are allowed
- Services include SSH/Ping/HTTP services

2. SSH Request Allowed

```
[khir@host1 ~]$ ssh root@192.168.29.248
The authenticity of host '192.168.29.248 (192.168.29.248)' can't be established.
ECDSA key fingerprint is SHA256:VuJfUuyBFBkXLNxUFZz8q9fwW+kU1XwR63jJ17cAKsU.
ECDSA key fingerprint is MD5:53:2d:69:e2:c9:bb:15:c1:9e:9e:18:e5:ba:97:06:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.29.248' (ECDSA) to the list of known hosts.
root@192.168.29.248's password:
Last login: Sat Apr 11 20:06:38 2020 from 192.168.50.3
[root@localhost ~]#
```

3. Ping Request Allowed

```
[khir@host1 ~]$ ping 192.168.29.248
PING 192.168.29.248 (192.168.29.248) 56(84) bytes of data.
64 bytes from 192.168.29.248: icmp_seq=1 ttl=64 time=3.60 ms
64 bytes from 192.168.29.248: icmp_seq=2 ttl=64 time=0.947 ms
64 bytes from 192.168.29.248: icmp_seq=3 ttl=64 time=0.951 ms
^C
--- 192.168.29.248 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.947/1.833/3.601/1.250 ms
[khir@host1 ~]$
[khir@host1 ~]$
```

# STRESS TEST OVERVIEW & RESULTS

# Test Overview – Stress Test

- **Test Case Objective:**
  - To verify the redundancy in the network infrastructure
- **Test Case Results:**
  - Verify the seamless operability of SDN infrastructure devices in case of different network failures
  - Validate the operability of backup solutions in the case of the Ryu SDN controller and Open vSwitch failures
- **Test Case Implication:**
  - The successful results sets base work towards the implementation on industry scale to achieve optimum network performance

# Test Cases and Strategy – Stress Test

| Test Case | Strategy | Expected Output | Result |
|---|---|---|---|
| SDN Controller redundancy in case of primary instance fail. | Verify if Docker swarm creates a new instance of the SDN controller if the existing instance fails. | “docker service ps ryunode” should display the active status of new instance spun and the shut down information of the failed containers. | PASS |
| OpenvSwitch instance creation if the existing OvS instance is shut. | Verify if Docker swarm creates a new instance of the OpenvSwitch if the existing instance fails. | “docker service ps ryunode” should display the active status of new instance spun and the shut down information of the failed containers. | PASS |

# Level 5: Stress Test for Network Devices

- **Redundancy Test – SDN Controller and Open vSwitch**

1. Docker Swarm Redundant Solution

- Docker swarm will maintain two SDN Controller instances on the two SDN servers
- If one instance fails, Docker swarm will spin a backup solution

2. Docker swarm status

```
[root@localhost ~]# docker service ps ryunode
ID                  NAME                IMAGE               NODE                    DESIRED STATE       CURRENT STATE           ERROR               PORTS
xafxld795ym4        ryunode.1           ryu312:latest       localhost.localdomain   Running             Running 2 months ago
lt5fria2mr0j        ryunode.2           ryu312:latest       localhost.localdomain   Running             Running 6 weeks ago
ntb86u59sq7k         \_ ryunode.2       ryu312:latest       localhost.localdomain   Shutdown            Shutdown 6 weeks ago
```

1. Docker Swarm Redundant Solution

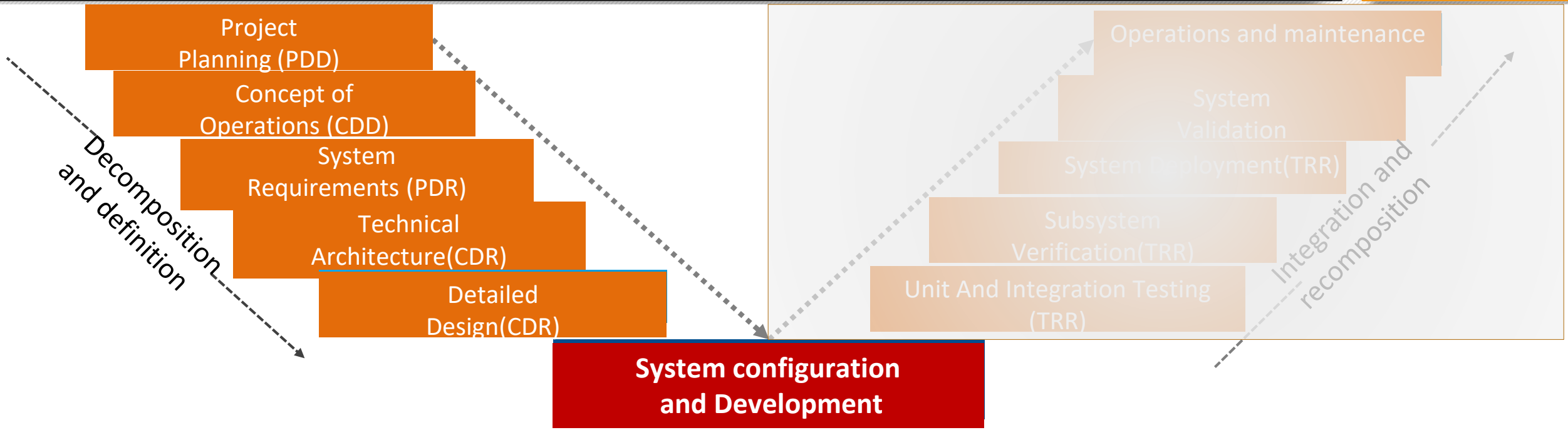
- Docker swarm will maintain one Open vSwitch instance on each hosts
- If one instance fails, Docker swarm will spin a backup solution

2. Docker swarm status

```
manesh@manesh-VirtualBox:~$ sudo docker service ps vswitch
ID                  NAME                IMAGE                           NODE                DESIRED STATE       CURRENT STATE           ERROR                         PORTS
qietgwt2b7vq        vswitch.1           globocom/openvswitch:latest     manesh-VirtualBox   Running             Running 1 second ago
us29ir7iqvl5         \_ vswitch.1       globocom/openvswitch:latest     manesh-VirtualBox   Shutdown            Failed 7 seconds ago    "task: non-zero exit (137)"
manesh@manesh-VirtualBox:~$
```

# SYSTEM ENGINEERING

# System Engineering “V” model - Fall



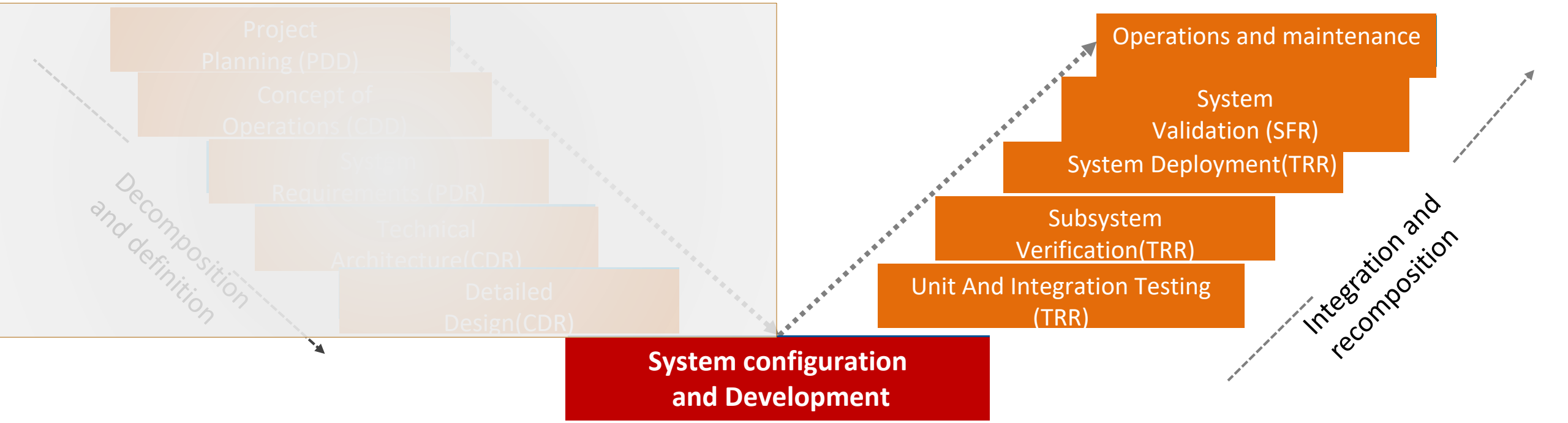

**Issues:**

- Finding simple backup solutions
- Needed to understand service chaining of VNFs

**Lessons Learnt:**

- Chose project elements based on budget, knowledge-base, user density and user requirements
- Understood the importance of documentation before implementation to understand requirements of each task

# System Engineering “V” model - Spring




## Issues:

- Lost physical accessibility to Dell Server due to VPN failure; changed testing strategy
- Formation of service chain using VNFs
- OpenStack version upgrade resulted in failure
- Due to knowledge gap in Java scripting, ONOS was replaced by RYU Controller

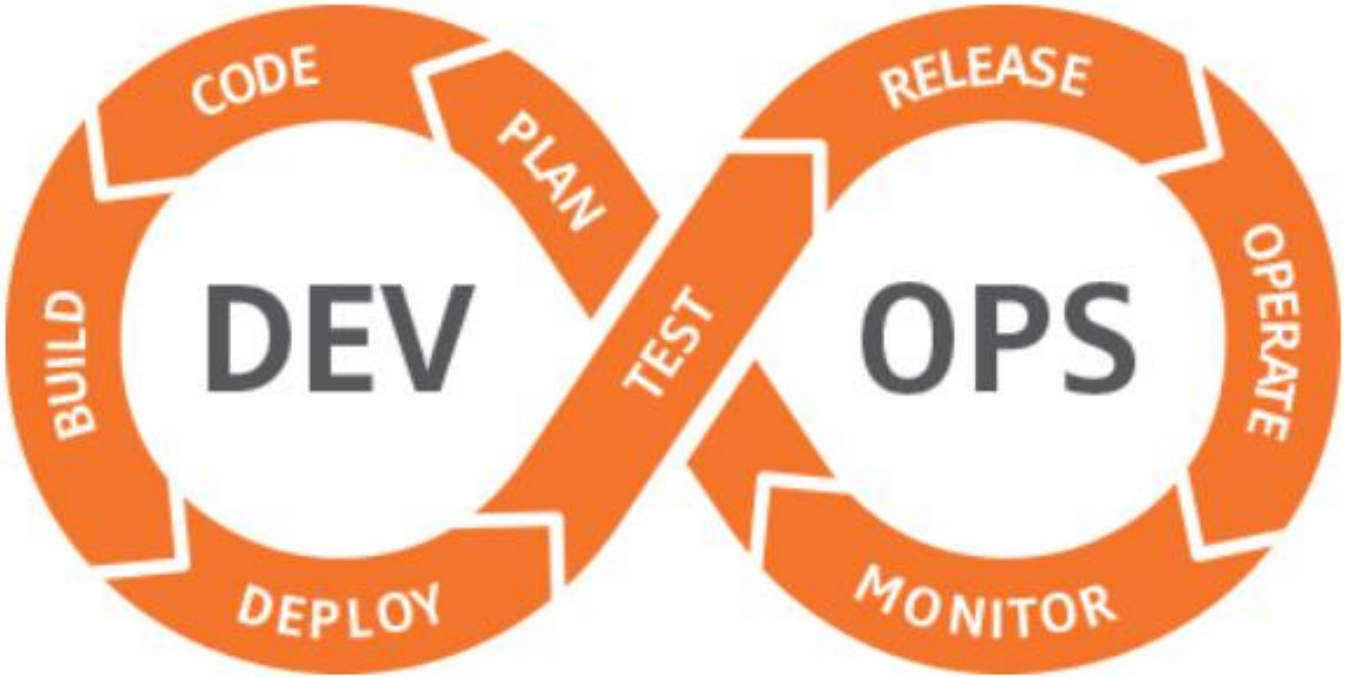
## Lessons Learnt:

- Understood the operation of Docker containers, Jenkin, OpenStack and DockerSwarm
- Learnt implementation of VNF and make sure they are interoperable
- Importance of multi-environment testing

# PROJECT MANAGEMENT

# Project Management

- DevOps approach for software development used
- Continuous integration and deployment that allows easy upgrades, high quality and reliable codes
- Testing for every change in code is automated in Jenkins. If successful, the code is deployed
- Packet analyzer on Test VM serves as the monitoring agent that provides continuous feedback
- The code is built in GitHub which has in-built version control



# Key Management Outcomes

## Successes

- Team Co-ordination: Discussions with team members for issue rectifications done remotely using Google Hangouts
- Periodic customer interactions: Weekly update from Faculty advisor for understanding the requirements granularly
- Records: All milestones, sub-tasks, progress and changes documented to the minute details

## Lessons Learnt

- Mutually exclusive tasks carried out parallelly and cross-verified by different members in the team
- Consider unforeseen issues and work under time constraints
- Keeping the design simple eliminates a lot of issues
- Understood the development of research paper in IEEE format

# Estimated Industry Cost

| | |
|---|---|
| Number of Engineers | 5 |
| Number of weeks | 28 |
| Hours/Week/Engineer | 18 |
| Cost/Hour | 30 |
| Total labor cost | $75,600 |
| Total cost with overhead rate (200%) | $151,200 |

# Q&A

# THANK YOU

# APPENDIX

# TEST OVERVIEW & RESULTS

# INTEGRATION TEST RESULTS

# Test Overview – Integration Test

- Test Case Objective:
  - To verify the connectivity between network elements
- Test Case Results:
  - The tests validate the peering between network devices and the intra-domain and internet connectivity of the network devices. The output verifies the accessibility of the VMs via SSH login and inter-VM reachability through the IP addresses allocated on the VM interfaces.
- Test Case Implication:
  - The results sets the base work for remote testing of the project solution. The basic reachability enables OpenFlow and REST API connectivity between network elements.

# Level 2: Integration of Network Infrastructure

- SDN infrastructure reachability

1. Intra-domain reachability

```
[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
```

2. Internet reachability

```
[root@localhost ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=16.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=18.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=22.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=54 time=15.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=54 time=10.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=54 time=14.8 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=54 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=54 time=22.7 ms
^C
--- 8.8.8.8 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7021ms
rtt min/avg/max/mdev = 10.721/17.135/22.985/3.887 ms
[root@localhost ~]#
```

3. OpenvSwitch and SDN Reachability

```
# ping 172.17.0.9
PING 172.17.0.9 (172.17.0.9): 56 data bytes
64 bytes from 172.17.0.9: seq=0 ttl=64 time=0.222 ms
64 bytes from 172.17.0.9: seq=1 ttl=64 time=0.239 ms
64 bytes from 172.17.0.9: seq=2 ttl=64 time=0.239 ms
64 bytes from 172.17.0.9: seq=3 ttl=64 time=0.239 ms
^C
--- 172.17.0.9 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.222/0.234/0.239 ms/
# ping 172.17.0.6
```

# Level 2: Integration of Network Infrastructure

- Bridge creation and bridge peering

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 2: Integration of Network Infrastructure

- VM and Container interface status

1. VM interface status

```
[root@localhost ~]# ip a | grep UP
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
5: enp0s10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
10: gre_sys@NONE: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65497 qdisc pfifo_fast master ovs-system state UNKNOWN group default qlen 1000
12: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
14: vethc627e3f@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
16: vetha85c46f@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
[root@localhost ~]#
```

2. OpenvSwitch interface status

```
/ # ip a | grep UP
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
22: eth1@if23: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
24: eth2@if25: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
26: eth3@if27: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
/ #
```

# COMPATIBILITY TEST RESULTS

# Test Overview – Compatibility Test

- Test Case Objective:
  - To verify the protocol level connectivity between network elements such as SDN and OpenvSwitch
- Test Case Results:
  - The tests verifies successful implementation of communications protocols using the Wireshark captures, the output of the curl command, device logs, and device status.
- Test Case Implication:
  - The results are crucial to enable decision making through SDN Controller by pushing reactive flows into the Openvswitch.

# Level 3: Compatibility between Network Devices

- OvS – SDN Controller OpenFlow connectivity

1. OpenvSwitch bridge connectivity status

```
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. Wireshark capture – OpenFlow connectivity

| | | | | | |
|---|---|---|---|---|---|
| 20 | 126.330166 | 192.168.122.38 | 192.168.122.244 | OpenFlow | 82 Type: OFPT_HELLO |
| 22 | 126.331884 | 192.168.122.244 | 192.168.122.38 | OpenFlow | 74 Type: OFPT_HELLO |
| 23 | 126.332001 | 192.168.122.244 | 192.168.122.38 | OpenFlow | 74 Type: OFPT_FEATURES_REQUEST |
| 26 | 126.333730 | 192.168.122.38 | 192.168.122.244 | OpenFlow | 98 Type: OFPT_FEATURES_REPLY |
| 27 | 126.334781 | 192.168.122.244 | 192.168.122.38 | OpenFlow | 82 Type: OFPT_MULTIPART_REQUEST, |
| 28 | 126.334890 | 192.168.122.244 | 192.168.122.38 | OpenFlow | 146 Type: OFPT_FLOW_MOD |

```
Frame 20: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
Ethernet II, Src: fe:a8:b3:09:80:30 (fe:a8:b3:09:80:30), Dst: PcsCompu_b3:47:10 (08:00:27:b3:47:10)
Internet Protocol Version 4, Src: 192.168.122.38, Dst: 192.168.122.244
Transmission Control Protocol, Src Port: 50926, Dst Port: 6653, Seq: 1, Ack: 1, Len: 16
OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_HELLO (0)
    Length: 16
    Transaction ID: 1
  Element
```

3. Github Pull/Push Connectivity

```
manesh@manesh-VirtualBox:~$ git remote add origin_team6 https://github.com/opennetworkinglab/OpenNetworkLinux
manesh@manesh-VirtualBox:~$ git pull origin_team6
warning: no common commits
remote: Enumerating objects: 35423, done.
remote: Total 35423 (delta 0), reused 0 (delta 0), pack-reused 35423
Receiving objects: 100% (35423/35423), 177.50 MiB | 26.29 MiB/s, done.
Resolving deltas: 100% (20700/20700), done.
From https://github.com/opennetworkinglab/OpenNetworkLinux
 * [new branch]      EC-ONLPv2-wedgebf-32x -> origin_team6/EC-ONLPv2-wedgebf-32x
 * [new branch]      ONLPv2     -> origin_team6/ONLPv2
 * [new branch]      ONLPv2-bazel -> origin_team6/ONLPv2-bazel
 * [new branch]      d10        -> origin_team6/d10
 * [new branch]      d5254-kernel-upgrade -> origin_team6/d5254-kernel-upgrade
 * [new branch]      docker-support -> origin_team6/docker-support
 * [new branch]      fix-692    -> origin_team6/fix-692
 * [new branch]      gh-pages   -> origin_team6/gh-pages
 * [new branch]      master     -> origin_team6/master
 * [new branch]      onf-ONLPv2 -> origin_team6/onf-ONLPv2
 * [new branch]      systemd    -> origin_team6/systemd
 * [new tag]         onlpv2-dev-1.1.0 -> onlpv2-dev-1.1.0
 * [new tag]         onlpv2-dev-1.1.1 -> onlpv2-dev-1.1.1
You asked to pull from the remote 'origin_team6', but did not specify
a branch. Because this is not the default configured remote
for your current branch, you must specify a branch on the command line.
manesh@manesh-VirtualBox:~$
```

```
manesh@manesh-VirtualBox:~$ git --version
git version 2.7.4
manesh@manesh-VirtualBox:~$
```

```
manesh@manesh-VirtualBox:~$ git config --list
user.name=maya8094
user.email=maya8094@colorado.edu
manesh@manesh-VirtualBox:~$
```

```
manesh@manesh-VirtualBox:~$ git add *
manesh@manesh-VirtualBox:~$ git branch
* master
manesh@manesh-VirtualBox:~$ git push team6_ori
Username for 'https://github.com': maya8094
Password for 'https://maya8094@github.com':
Everything up-to-date
manesh@manesh-VirtualBox:~$
```

# PERFORMANCE TEST RESULTS

# Test Cases and Strategy – Performance Test

| Test Case | Strategy | Expected Output | Result |
|---|---|---|---|
| NAT | Verify source IP Address translation from private to public IP Address | Wireshark capture displays the public IP address as source ip address | PASS |
| L2 Forwarding | Verify basic forwarding of all packets | Wireshark capture displays the normal exit of packets from the interface without modification | PASS |
| L3 Routing | Verify the forwarding of inter-domain traffic to the default gateway | Wireshark capture displays the packet reception at the default gateway IP | PASS |

# Level 4: Performance Test for VNFs

- Performance Test – NAT Setup and Functionality

1. VNF Creation – NAT (Flow Table 3 – Vswitch1)

- For public domain reachability, the private IP address assigned needs to be translated to the public IP address
- **Packets originated by the host:** Host IP address is replaced with the NAT assigned IP Address before forwarding the packets
- **Packets received by the host:** The destination IP address is translated back to the host IP address before forwarding the packet to the kernel
- Complete internet reachability was ensured

```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=39.325s, table=0, n_packets=0, n_bytes=0, idle_timeout=64,
 idle_age=39, priority=100,tcp,in_port=1,nw_src=192.168.122.38,tp_src=52925,tp_d
st=8000 actions=mod_nw_src:192.168.122.2,mod_tp_src:59999,output:3
```

2. Internet reachability

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=53 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=53 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=53 time=12.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=53 time=13.1 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=53 time=11.8 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 11.874/14.187/17.578/2.085 ms
```

# Level 4: Performance Test for VNFs

- Performance Test – Layer 2 Setup and Functionality

1. VNF Creation – L2 (Flow Table 0 – Vswitch1 – Host2)

- Layer2 VNF is enabled on Host2. Request coming on one port will be forwarded to the other port by OvSwitch
- Both intra-domain and inter-domain requests will be forwarded to the ingress or egress interfaces
- No packet filtering or special treatment given
- Both kind of domain reachability is successful

```
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x64, duration=3883316.946s, table=0, n_packets=17653142, n_bytes=3002090444, priority=1200,in_port=2 actions=goto_table:1
 cookie=0x65, duration=3883316.939s, table=0, n_packets=59682, n_bytes=3955982, priority=1201,in_port=1 actions=goto_table:1
 cookie=0xcc, duration=3883316.898s, table=1, n_packets=59671, n_bytes=3955028, priority=10,in_port=1 actions=output:2
 cookie=0xce, duration=3883316.075s, table=1, n_packets=17653124, n_bytes=3002088694, priority=10,in_port=2 actions=output:1
```

2. Intra-domain reachability

```
[root@host2 ~]# ping 192.168.122.38
PING 192.168.122.38 (192.168.122.38) 56(84) bytes of data.
64 bytes from 192.168.122.38: icmp_seq=1 ttl=64 time=5.47 ms
64 bytes from 192.168.122.38: icmp_seq=2 ttl=64 time=1.41 ms
^C
--- 192.168.122.38 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.410/3.440/5.471/2.031 ms
[root@host2 ~]#
```

2. Inter-domain reachability

```
[root@host2 ~]# ping 192.168.29.68
PING 192.168.29.68 (192.168.29.68) 56(84) bytes of data.
64 bytes from 192.168.29.68: icmp_seq=1 ttl=64 time=5.30 ms
64 bytes from 192.168.29.68: icmp_seq=2 ttl=64 time=1.16 ms
^C
--- 192.168.29.68 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.166/3.233/5.300/2.067 ms
[root@host2 ~]#
```

# Level 4: Performance Test for VNFs

- Performance Test – Layer 3 Setup and Functionality

1. VNF Creation – L3 (Flow Table 2 – Vswitch1 – Host2)

- Granular implementation of flows for specific inter-domain IP Addresses
- Flows related to outgoing and incoming requests on the hosts are added
- Inter-domain reachability requests were successful for both incoming and outgoing requests

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data | grep table=1 | grep output
 cookie=0xf9, duration=3873047.706s, table=1, n_packets=24320, n_bytes=2615215, priority=144,ip,in_port=2,nw_dst=192.168.122.38 actions=output:1
 cookie=0xfa, duration=3873047.699s, table=1, n_packets=7915100, n_bytes=474906000, priority=145,arp,in_port=2,arp_tpa=192.168.122.38 actions=output:1
 cookie=0xf7, duration=3873047.692s, table=1, n_packets=7530, n_bytes=610010, priority=147,ip,in_port=1,nw_dst=192.168.29.7 actions=output:2
 cookie=0xf8, duration=3873047.685s, table=1, n_packets=47601, n_bytes=1999242, priority=148,arp,in_port=1,arp_tpa=192.168.29.7 actions=output:2
 cookie=0x105, duration=3873040.494s, table=1, n_packets=0, n_bytes=0, priority=161,ip,in_port=2,nw_dst=192.168.122.1 actions=output:1
 cookie=0x106, duration=3873040.487s, table=1, n_packets=1730803, n_bytes=72693924, priority=162,arp,in_port=2,arp_tpa=192.168.122.1 actions=output:1
 cookie=0x11f, duration=3873047.677s, table=1, n_packets=4, n_bytes=392, priority=147,ip,in_port=2,nw_src=192.168.29.7 actions=output:1
 cookie=0x120, duration=3873046.988s, table=1, n_packets=15408, n_bytes=924480, priority=148,arp,in_port=2,arp_spa=192.168.29.7 actions=output:1
 cookie=0xf4, duration=3873047.722s, table=1, n_packets=7235147, n_bytes=310706256, priority=143,arp,in_port=1,arp_tpa=192.168.122.0/24 actions=output:2
 cookie=0xf5, duration=3873047.712s, table=1, n_packets=8434, n_bytes=752021, priority=146,ip,in_port=1,nw_dst=192.168.122.0/24 actions=output:2
/ #
```

2. Inter-domain ping request

```
[khir@host1 ~]$ ping 192.168.29.248
PING 192.168.29.248 (192.168.29.248) 56(84) bytes of data.
64 bytes from 192.168.29.248: icmp_seq=1 ttl=64 time=3.60 ms
64 bytes from 192.168.29.248: icmp_seq=2 ttl=64 time=0.947 ms
64 bytes from 192.168.29.248: icmp_seq=3 ttl=64 time=0.951 ms
^C
--- 192.168.29.248 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.947/1.833/3.601/1.250 ms
[khir@host1 ~]$
[khir@host1 ~]$
```

3. Inter-domain SSH request

```
[khir@host1 ~]$ ssh root@192.168.29.248
The authenticity of host '192.168.29.248 (192.168.29.248)' can't be established.
ECDSA key fingerprint is SHA256:VuJfUuyBFBkXLNxUFZz8q9fwW+kU1XwR63jJ17cAKsU.
ECDSA key fingerprint is MD5:53:2d:69:e2:c9:bb:15:c1:9e:9e:18:e5:ba:97:06:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.29.248' (ECDSA) to the list of known hosts.
root@192.168.29.248's password:
Last login: Sat Apr 11 20:06:38 2020 from 192.168.50.3
[root@localhost ~]#
```

# MULTI-ENVIRONMENT TEST RESULTS

# Test Overview – Multienvironment Test

- Test Case Objective:
  - To verify the operability of solution on different OS.
- Test Case Results:
  - The tests verifies the operability and functionality of VNFs and ensures seamless functionality of service chain in any host environment.
- Test Case Implication:
  - The results ensures the scope of the solution as a lightweight, host OS independent and robust network application

# Level 6: Multi-environment Network Deployment

- Solution deployment on MacOS - Deployment

1. Docker installation

```
Maneshs-MacBook-Pro:~ maya$ docker version
Client:
 Version:      17.03.1-ce-rc1
 API version:  1.27
 Go version:   go1.7.5
 Git commit:   3476dbf
 Built:        Fri Mar 17 00:27:41 2017
 OS/Arch:      darwin/amd64

Server:
 Version:      17.03.1-ce-rc1
 API version:  1.27 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   3476dbf
 Built:        Wed Mar 15 20:28:18 2017
 OS/Arch:      linux/amd64
 Experimental: true
```

2. OpenvSwitch installation

```
Maneshs-MacBook-Pro:~ maya$ docker run -it --rm --name vswitch --cap-add=NET_ADMIN -d globocom/openvswitch
Unable to find image 'globocom/openvswitch:latest' locally
latest: Pulling from globocom/openvswitch
ff3a5c916c92: Pull complete
a1e2db7c9db6: Pull complete
a477371faa9b: Pull complete
1ff1c88b1fca: Pull complete
b04e3c06a506: Pull complete
24a45206440d: Pull complete
6bd0e8a2022e: Pull complete
31354961aae9: Pull complete
Digest: sha256:bce391647b461da6cbd2f7e566d1df24995bc3030f11cdb48e206e2b34d67c45
Status: Downloaded newer image for globocom/openvswitch:latest
09a3746e5af2382f1c1813854952f36a9039c9e7b78386276a07a360e5244a23
```

3. OpenvSwitch Status

```
Maneshs-MacBook-Pro:~ maya$ docker ps -a
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS
              NAMES
09a3746e5af2        globocom/openvswitch   "/bin/sh -c /usr/b..."   4 minutes ago       Up 4 minutes        22/tcp
              vswitch
Maneshs-MacBook-Pro:~ maya$
```

# Level 6: Multi-environment Network Deployment

- Solution deployment on MacOS – Bridge creation and peering

1. Docker bridge Creation

```
Maneshs-MacBook-Pro:~ maya$ docker network create -d bridge capstone-in
3adebf32b2825bdf95d00af804c7e1e885e5d107c8ffc31f7255b6622ec75e5e
Maneshs-MacBook-Pro:~ maya$ docker network create -d bridge capstone-out
239bc5f32fdf1ee5927d00d9fa8e0fe2843639a8595ae52e89a6b11d7839116b
Maneshs-MacBook-Pro:~ maya$
```

2. OpenvSwitch bridge peering

```
Maneshs-MacBook-Pro:~ maya$ docker network connect capstone-in vswitch
Maneshs-MacBook-Pro:~ maya$ docker network connect capstone-out vswitch
Maneshs-MacBook-Pro:~ maya$
```

3. OpenvSwitch interface Status

```
Maneshs-MacBook-Pro:~ maya$ docker exec -it vswitch /bin/sh
/ # ip a | grep UP
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
21: eth1@if22: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue master ovs-system state UP
23: eth2@if24: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue master ovs-system state UP
/ #
```

```
Maneshs-MacBook-Pro:~ maya$ docker exec -it vswitch /bin/sh
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:728 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25348 (24.7 KiB)  TX bytes:2164 (2.1 KiB)

eth1      Link encap:Ethernet  HWaddr 02:42:AC:14:00:02
          inet addr:172.20.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe14:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1296 (1.2 KiB)  TX bytes:648 (648.0 B)

eth2      Link encap:Ethernet  HWaddr 02:42:AC:15:00:02
          inet addr:172.21.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe15:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1296 (1.2 KiB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

# Level 6: Multi-environment Network Deployment

- Solution deployment on MacOS – SDN network setup

1. SDN bridge and controller connectivity

```
/ # ovs-vsctl show
ec531027-1153-41f4-bd5c-f0d347997dd4
    Bridge br-data
        Controller "tcp:192.168.122.244:6633"
        Port "eth2"
            Interface "eth2"
        Port "eth1"
            Interface "eth1"
        Port br-data
            Interface br-data
                type: internal
/ #
```

2. OpenvSwitch protocol setup

```
/ # ovs-vsctl add-br br-data
/ # ovs-vsctl set bridge br-data protocols=OpenFlow13
/ # ifconfig eth1 up
/ # ifconfig eth2 up
/ # ovs-vsctl add-port br-data eth1
/ # ovs-vsctl add-port br-data eth2
/ # ovs-vsctl set-controller br-data tcp:192.168.122.117:6633
/ # ovs-vsctl set-controller br-data tcp:192.168.122.244:6633
```

3. OpenvSwitch flow table

```
/ # ovs-ofctl dump-flows -O OpenFlow13 br-data
 cookie=0xcc, duration=69.892s, table=0, n_packets=0, n_bytes=0, priority=103,in_port=eth1 actions=output:eth2
 cookie=0xce, duration=65.304s, table=0, n_packets=0, n_bytes=0, priority=105,in_port=eth2 actions=output:eth1
 cookie=0x67, duration=50.857s, table=0, n_packets=0, n_bytes=0, priority=103,in_port=eth2,dl_vlan=6 actions=pop_vlan,output:eth1
 cookie=0x68, duration=46.605s, table=0, n_packets=0, n_bytes=0, priority=104,in_port=eth2,dl_vlan=5 actions=pop_vlan,output:eth1
 cookie=0x65, duration=50.867s, table=0, n_packets=0, n_bytes=0, priority=101,in_port=eth1 actions=push_vlan:0x8100,output:6,output:eth2
 cookie=0x66, duration=50.862s, table=0, n_packets=0, n_bytes=0, priority=102,in_port=eth1 actions=push_vlan:0x8100,output:5,output:eth2
/ #
```

# Level 6: Multi-environment Network Deployment

- Solution deployment on MacOS – Intra-domain and inter-domain reachability

1. Intra-domain reachability

```
Maneshs-MacBook-Pro:~ maya$ ping 192.168.122.244
PING 192.168.122.244 (192.168.122.244): 56 data bytes
64 bytes from 192.168.122.244: seq=0 ttl=64 time=0.512 ms
64 bytes from 192.168.122.244: seq=1 ttl=64 time=0.549 ms
64 bytes from 192.168.122.244: seq=2 ttl=64 time=0.551 ms
64 bytes from 192.168.122.244: seq=3 ttl=64 time=0.573 ms
^C
--- 192.168.122.244 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.512/0.544/0.569 ms
```

2. Inter-domain reachability

```
Maneshs-MacBook-Pro:~ maya$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=54 time=20.126 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=19.559 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=14.638 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=11.543 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=54 time=12.809 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=54 time=14.160 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=54 time=12.474 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 11.543/15.044/20.126/3.185 ms
Maneshs-MacBook-Pro:~ maya$
```

# AUTOMATION TEST RESULTS

# Automation Test Results

## 1. Deployment Test Automation Result

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.DeploymentTest
********EXECUTING DEPLOYMENT TESTS********* ... ok
opsdeploy_test (CNIFH.DeploymentTest) ... ok
ops_vm_creation_test (CNIFH.DeploymentTest) ... ok
ops_vm_ip_chk_test (CNIFH.DeploymentTest) ... ok
ops_vm_vitualbri_chk_test (CNIFH.DeploymentTest) ... ok
docker_install_test (CNIFH.DeploymentTest) ... ok
controller1_install_test (CNIFH.DeploymentTest) ... ok
controller2_install_test (CNIFH.DeploymentTest) ... ok
host1_ovs_install_test (CNIFH.DeploymentTest) ... ok
host2_ovs_install_test (CNIFH.DeploymentTest) ... ok
test_vm_install_test (CNIFH.DeploymentTest) ... ok
host1_ovs1_flowtable_chk_test (CNIFH.DeploymentTest) ... ok
host1_ovs2_flowtable_chk_test (CNIFH.DeploymentTest) ... ok
host2_ovs1_flowtable_chk_test (CNIFH.DeploymentTest) ... ok

----------------------------------------------------------------------
Ran 14 tests in 0.001s

OK
```

## 2. Integration Test Automation Result

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.IntegrationTest
********EXECUTING INTEGRATION TESTS********* ... ok
vms_conn_test (CNIFH.IntegrationTest) ... ok
host1_ovs_bridge_test (CNIFH.IntegrationTest) ... ok
conn_ovs_sdn_test (CNIFH.IntegrationTest) ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

## 3. Compatibility Test Automation Result

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.CompatibilityTest
********EXECUTING COMPATIBILITY TESTS********* ... ok
of_ovs_contrl_test (CNIFH.CompatibilityTest) ... ok
restapi_chk_test (CNIFH.CompatibilityTest) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

## 4. Stress Test Automation Result

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.StressTest
********EXECUTING REDUNDANCY TESTS********* ... ok
cntrl_redundancy_test (CNIFH.StressTest) ... ok
ovs_redundancy_test (CNIFH.StressTest) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

# Automation Test Results

## 1. Performance Test Automation Result - VLAN

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.VlanTest
********EXECUTING VLAN PERFORMANCE TESTS********* ... ok
vlan_tag_chk_test (CNIFH.VlanTest) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

## 2. Performance Test Automation Result - QoS

```
OK
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.QOSTest
********EXECUTING QOS PERFORMANCE TESTS********* ... ok
qos_perf_chk_test (CNIFH.QOSTest) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

## 3. Performance Test Automation Result - Firewall

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.FirewallPerformanceTest

********EXECUTING FIREWALL PERFORMANCE TESTS********* ... ok
ssh_blocked_test (CNIFH.FirewallPerformanceTest) ... ok
ssh_allowed_test (CNIFH.FirewallPerformanceTest) ... ok
ping_block_test (CNIFH.FirewallPerformanceTest) ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

## 4. Performance Test Automation Result - ARP

```
(venv) Afures-MacBook-Pro:FinalProject afuremarthaoyibo$ nosetests -v CNIFH.L2_L3_ARP_test
********EXECUTING L2_L3_ARP PERFORMANCE TESTS********* ... ok
intra_domain_conn_test (CNIFH.L2_L3_ARP_test) ... ok
inter_domain_conn_test (CNIFH.L2_L3_ARP_test) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```