

Critical Design 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.
Problem with Traditional Networking	Soumya	3
Benefits of SDN-NFV using Containers	Soumya	4
Project Objectives	Soumya	5
FBD and CONOPS	Manesh	6 - 7
Levels of Success	Manesh	8
Critical Project Elements	Manesh	9 - 11
Design Requirements	Manesh	12 - 17
Baseline Design Summary	Kiran	18 - 21
Verification & Validation	Kiran	22 - 34
Risk Factors & their Impact	Prarthana	35 - 41
Project Risks	Prarthana	42 - 44
Revised CONOPS	Soumya	45
Organizational Chart	Soumya	46
Work Breakdown Structure	Soumya	47
Cost Plan	Martha	48
Hardware Requirements	Martha	49
Project Targets & Key Deadlines	Martha	50

Problem - Traditional Networking



Difficult to deploy and scale



Lacks flexibility



Large CAPEX and OPEX



Failures and packet losses



Administrative overhead



Error-prone

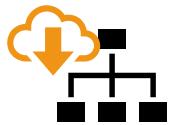


Network complexity



Short product life cycle

Benefits- SDN NFV using Containers



Quick deployment



Reduced CAPEX and OPEX



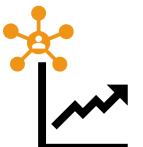
Reduced kernel & administrative overhead



Simplified network architecture



Flexibility and scalability



Proactive failure detection



NSOT



Timely updates

Project Objectives

Objective

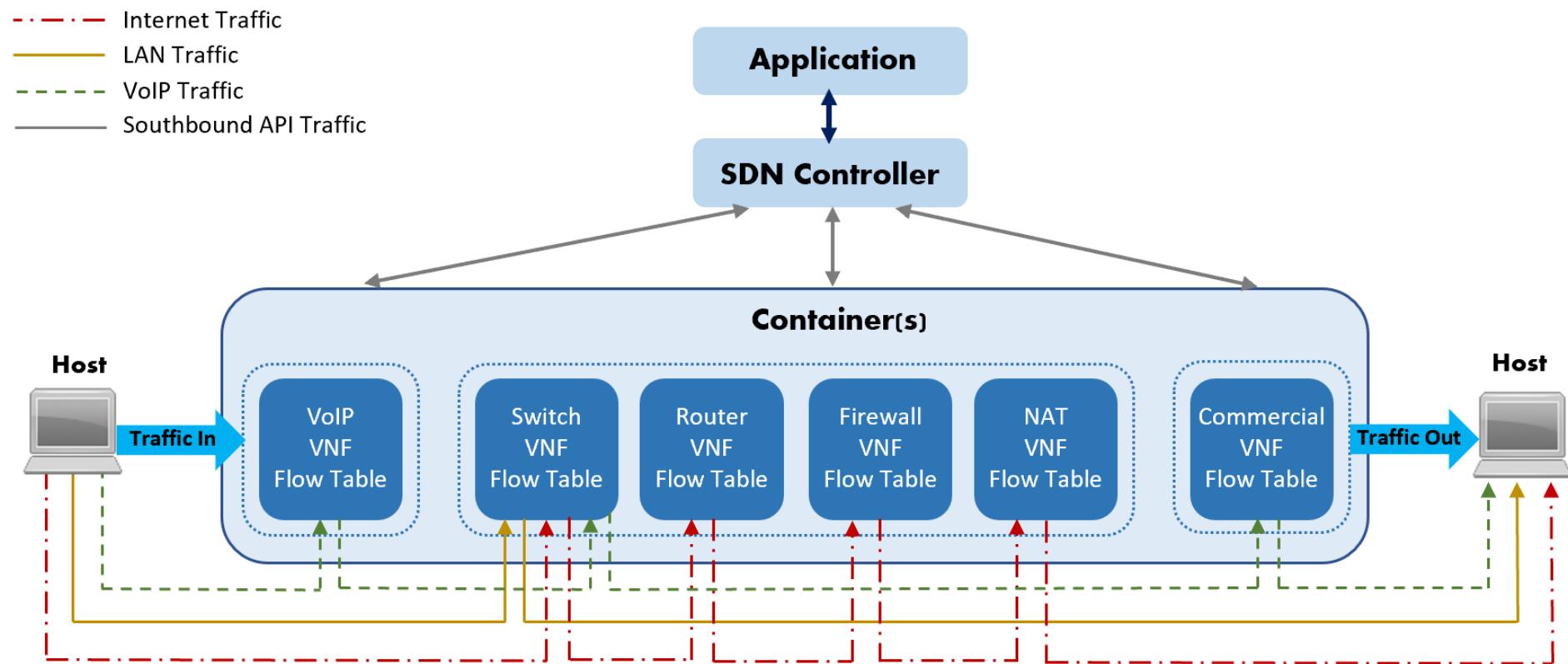
- Create a low-cost solution for the deployment of SDN based network services such as routers, firewalls, VoIP, etc. on the host devices through NFV:
 - NFV based services will be deployed through container based VNFs
 - Enhanced performance by bringing SDN based intelligence closer to the end hosts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



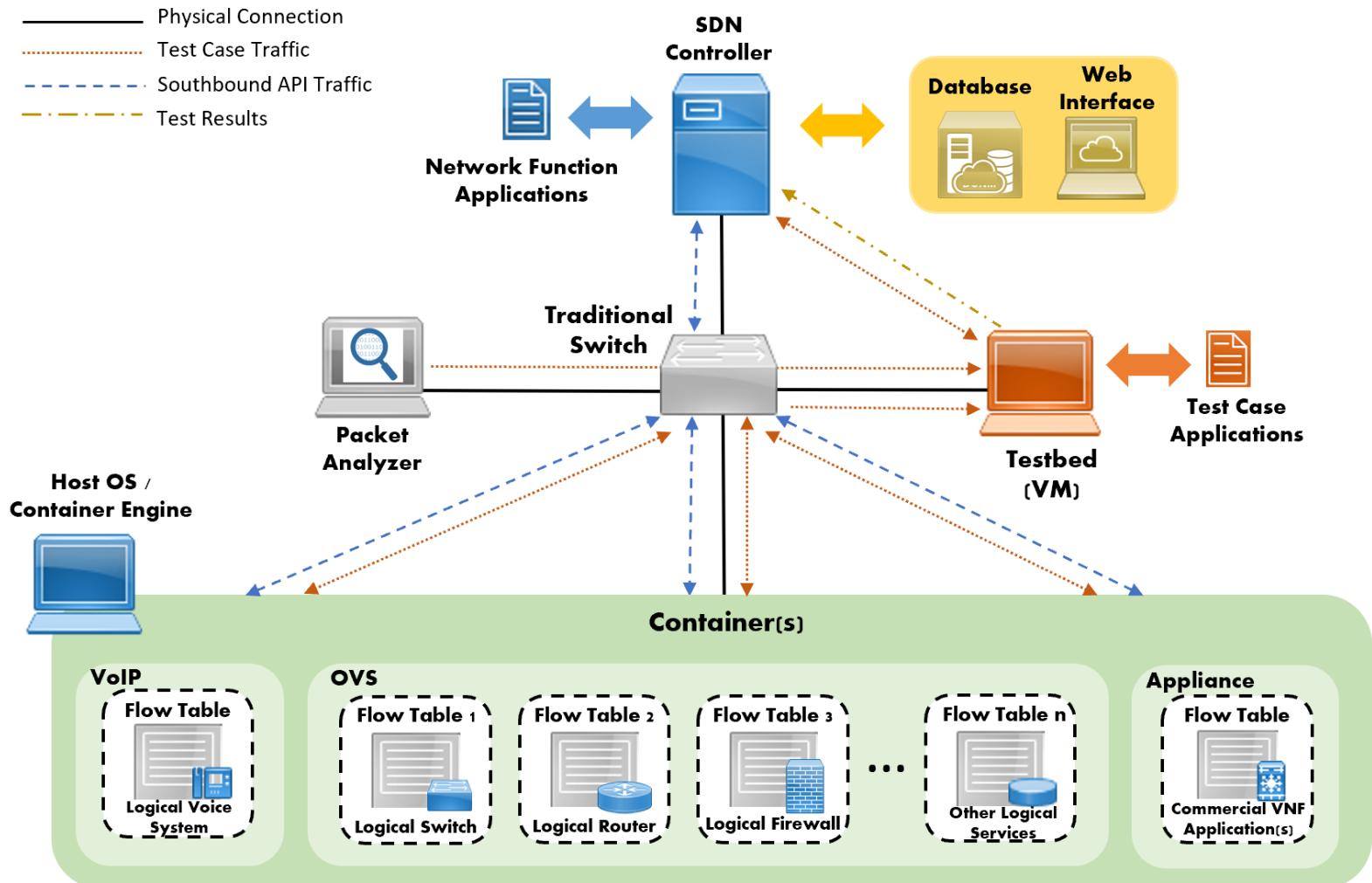
Stretch Goal

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

Functional Block Diagram (FBD)



Concept of Operations (CONOPS)



Levels of Success

Level 1: Configuration Tests

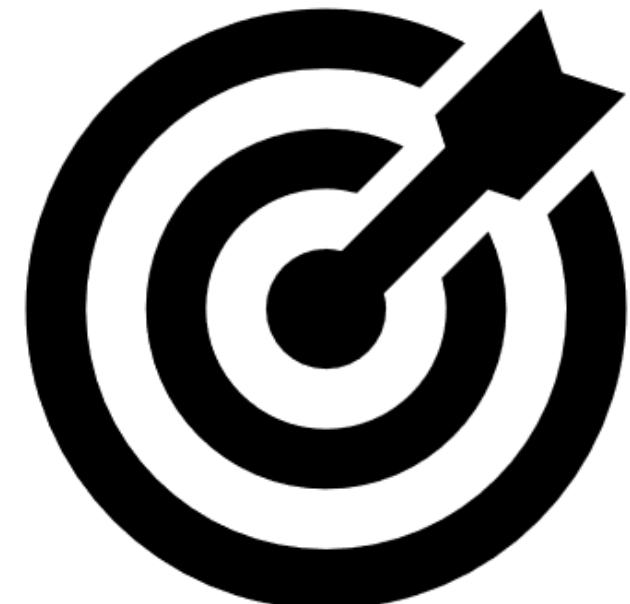
- Test the deployment of VNFs
- Test the service chain creation
- Test the reachability between network devices

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



Critical Project Elements - Technical

CPE 1.1:
VNF Creation

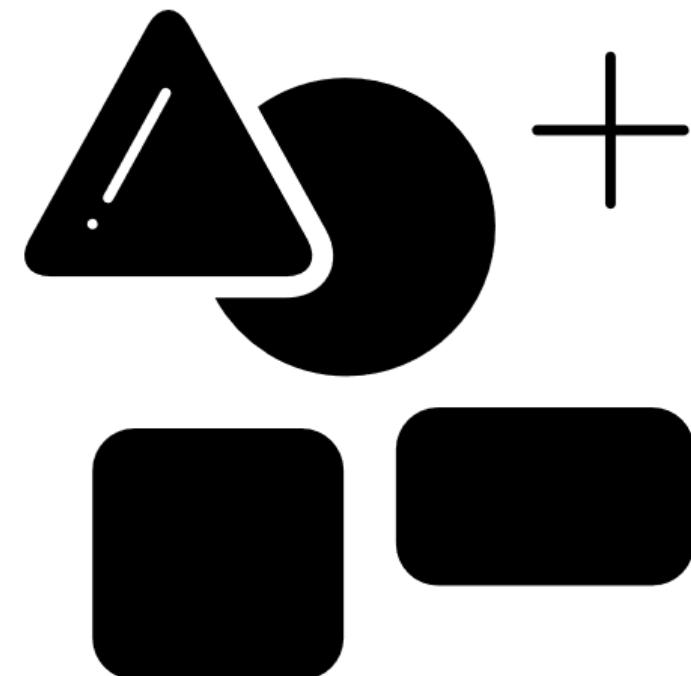
- Deploy OVS Switch using containers
- Create VNFs corresponding to network services on OVS Switches

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



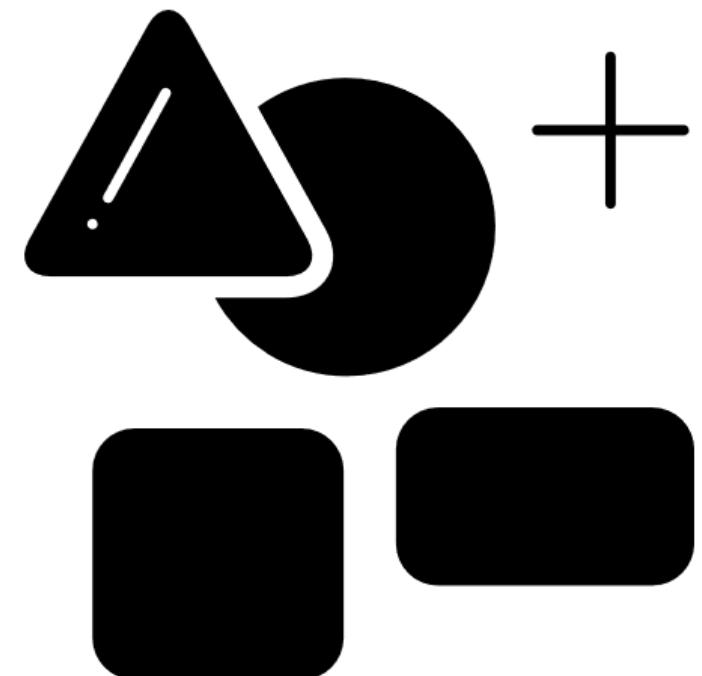
Critical Project Elements - Technical (continued)

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

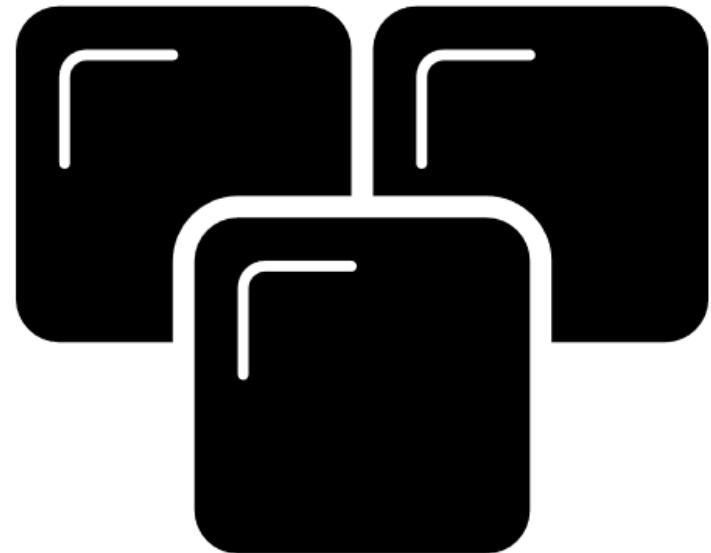
- Using servers and traditional switches available in Telecommunications lab
- Laptops with Windows, Linux and MacOS as host devices

CPE 2.2:
Containers/
Controllers

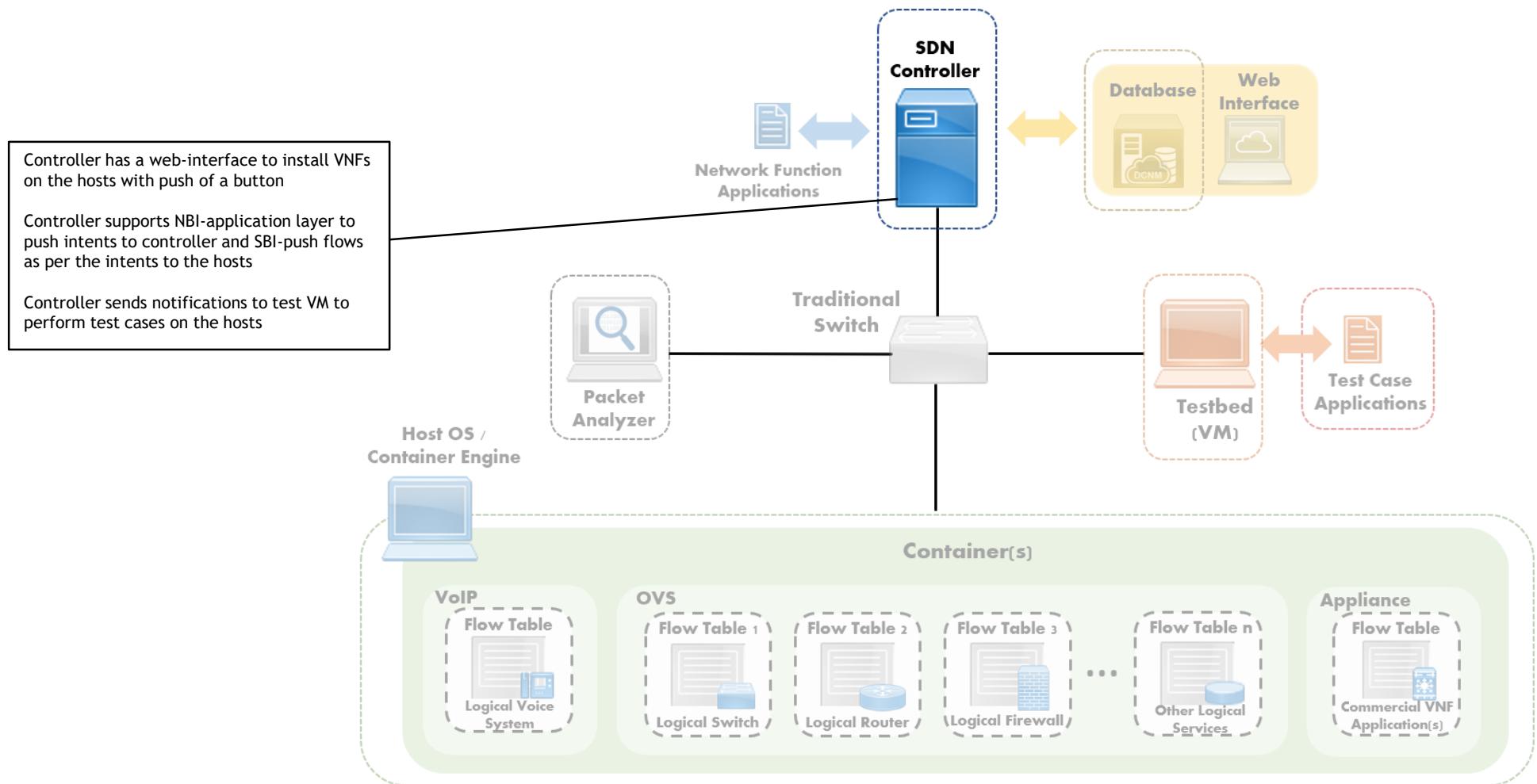
- Using open-source software for topology creation
- Dockers as containers, ONOS as SDN controller

CPE 2.3:
Knowledge and
concepts

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

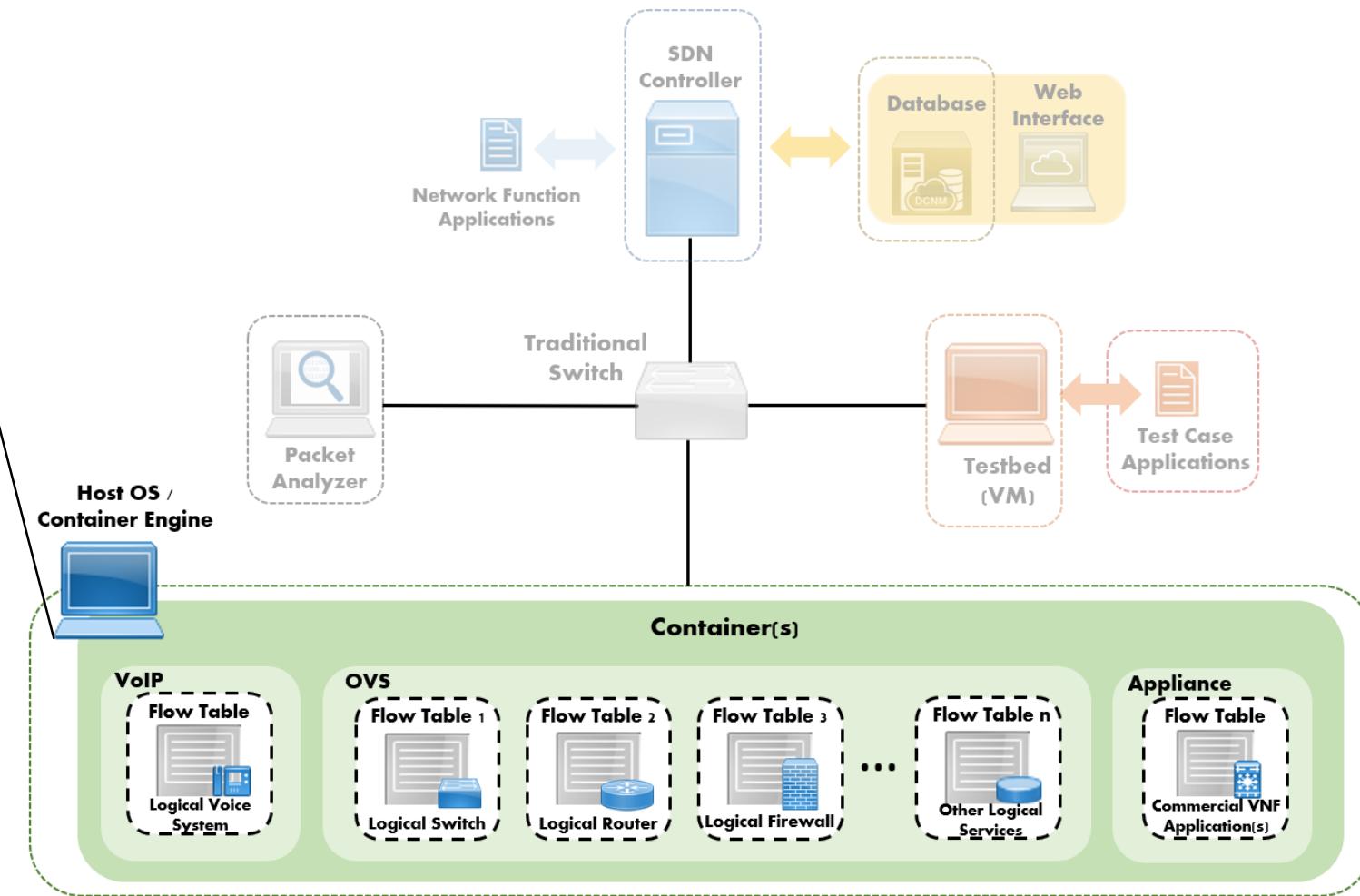


Design Requirements

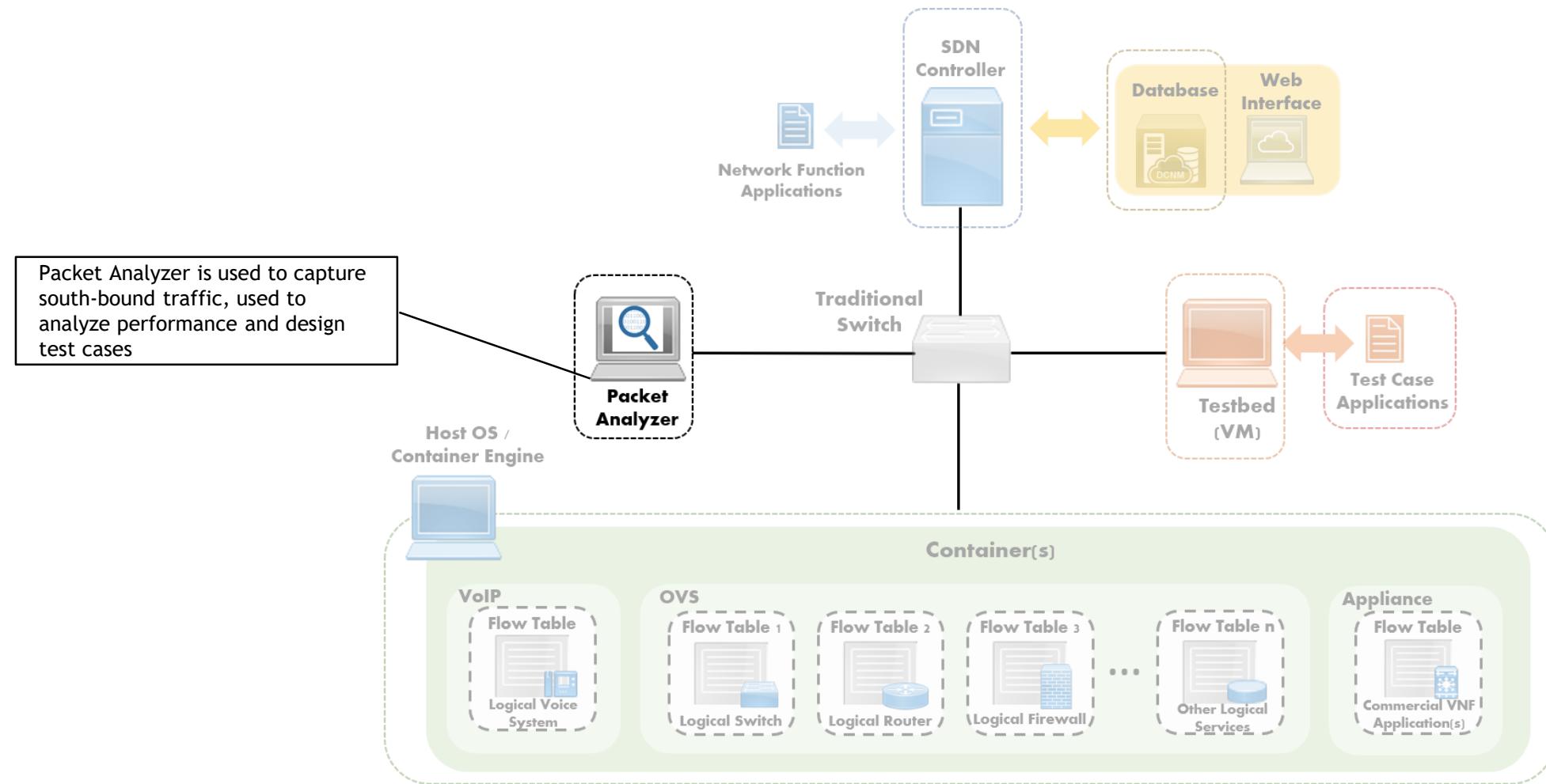


Design Requirements (continued)

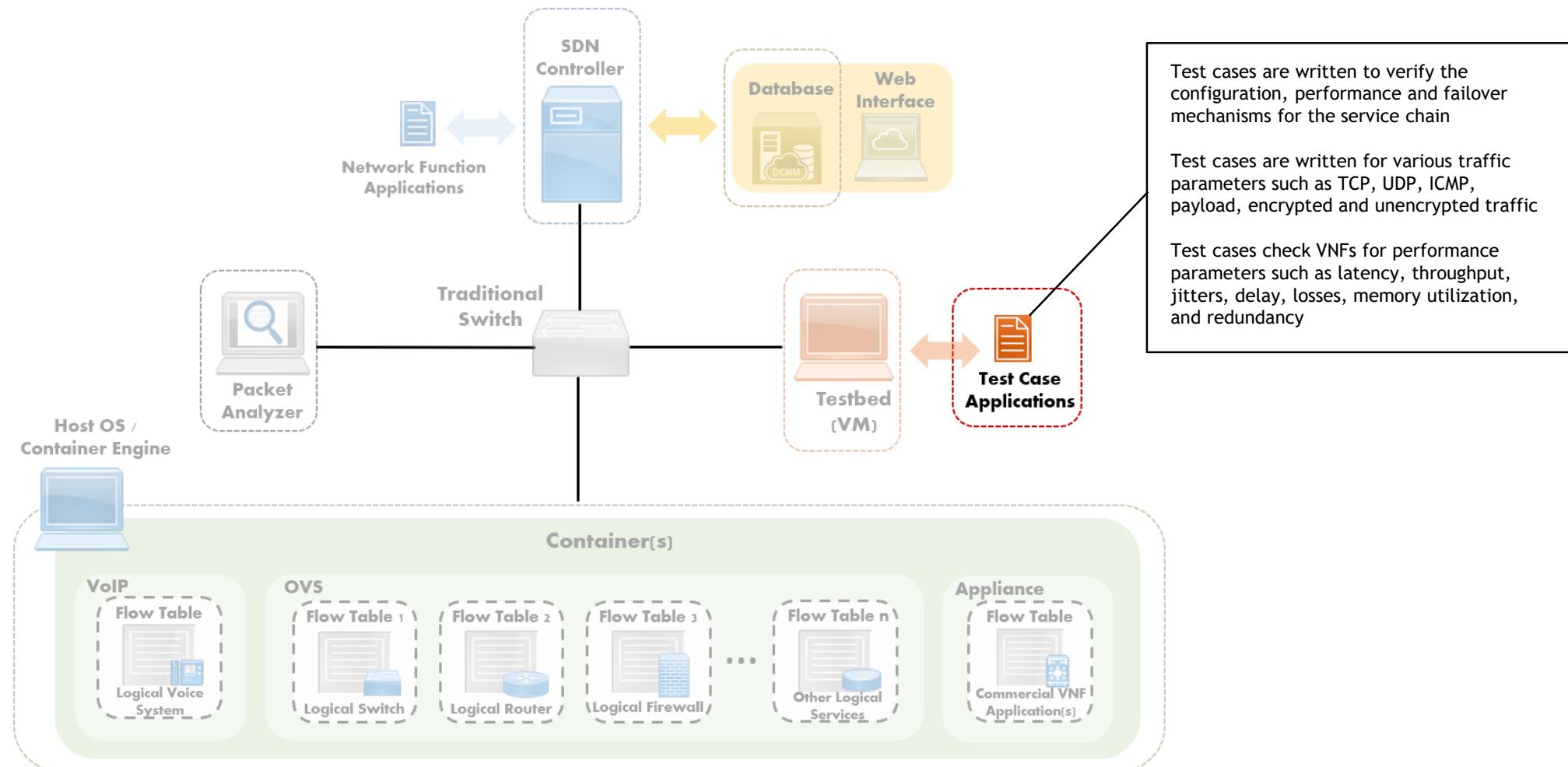
Containers are hosted on Windows and Linux OS
Containers are used to spin up VNFs
OVS images are used to create VNFs
VNFs are created for individual components in the service chain



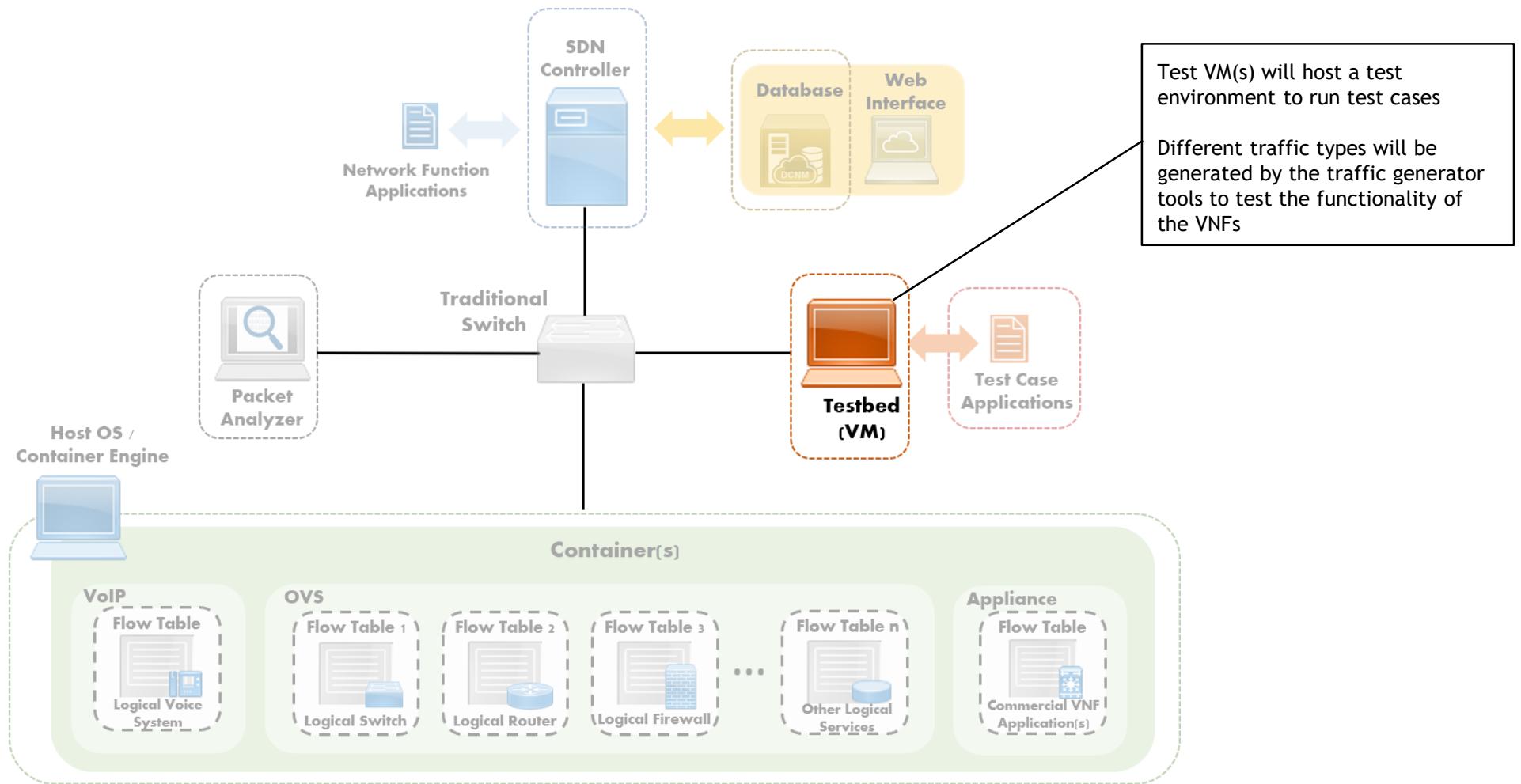
Design Requirements (continued)



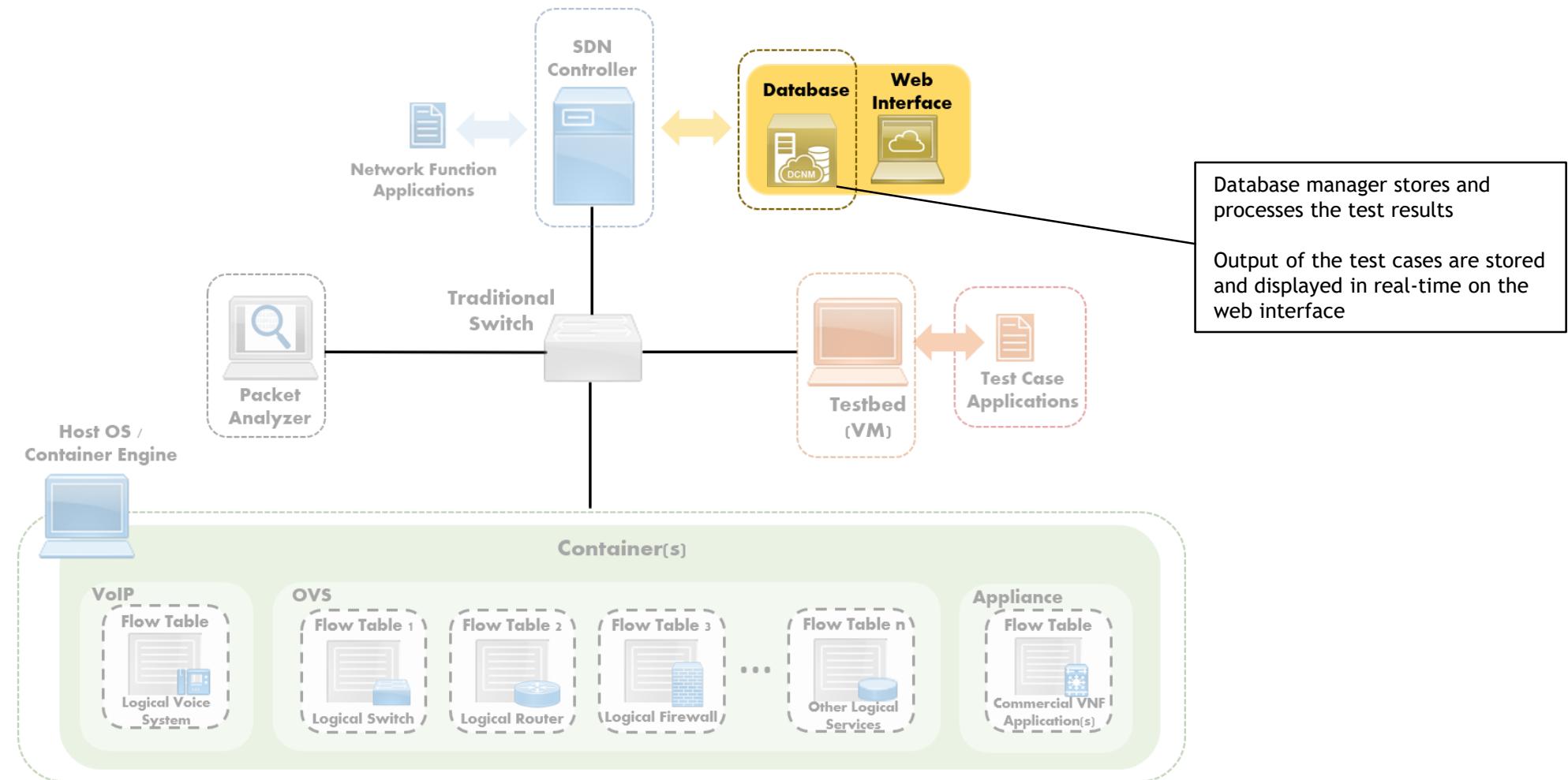
Design Requirements (continued)



Design Requirements (continued)



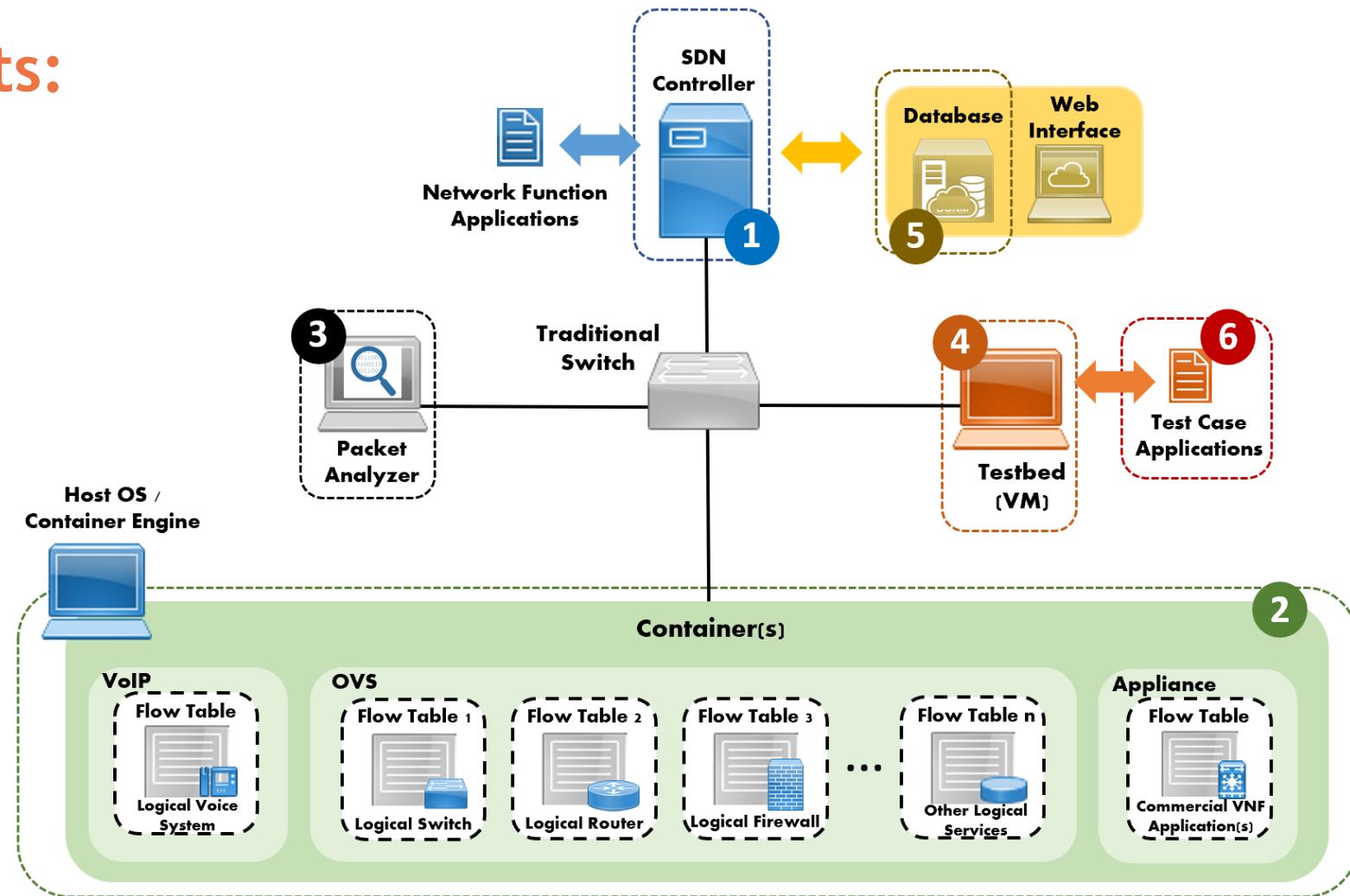
Design Requirements (continued)



Baseline Design from CDD

Key Design Elements:

- 1 SDN Controller
- 2 Containers
- 3 Packet Analyzer
- 4 Hypervisor
- 5 Database manager
- 6 Traffic generator



Baseline Design Summary

Project Elements	Selected Element	Rationale
SDN Controller	ONOS	Provides northbound abstraction using REST Supports multiple southbound protocols Well documented Interactive GUI
Container	Docker	Lightweight Can be deployed on Linux and Windows OS Suitable for continuous deployment and integration
Hypervisor	KVM	Enhanced VM security and isolation Bare metal speed Better network virtualization support

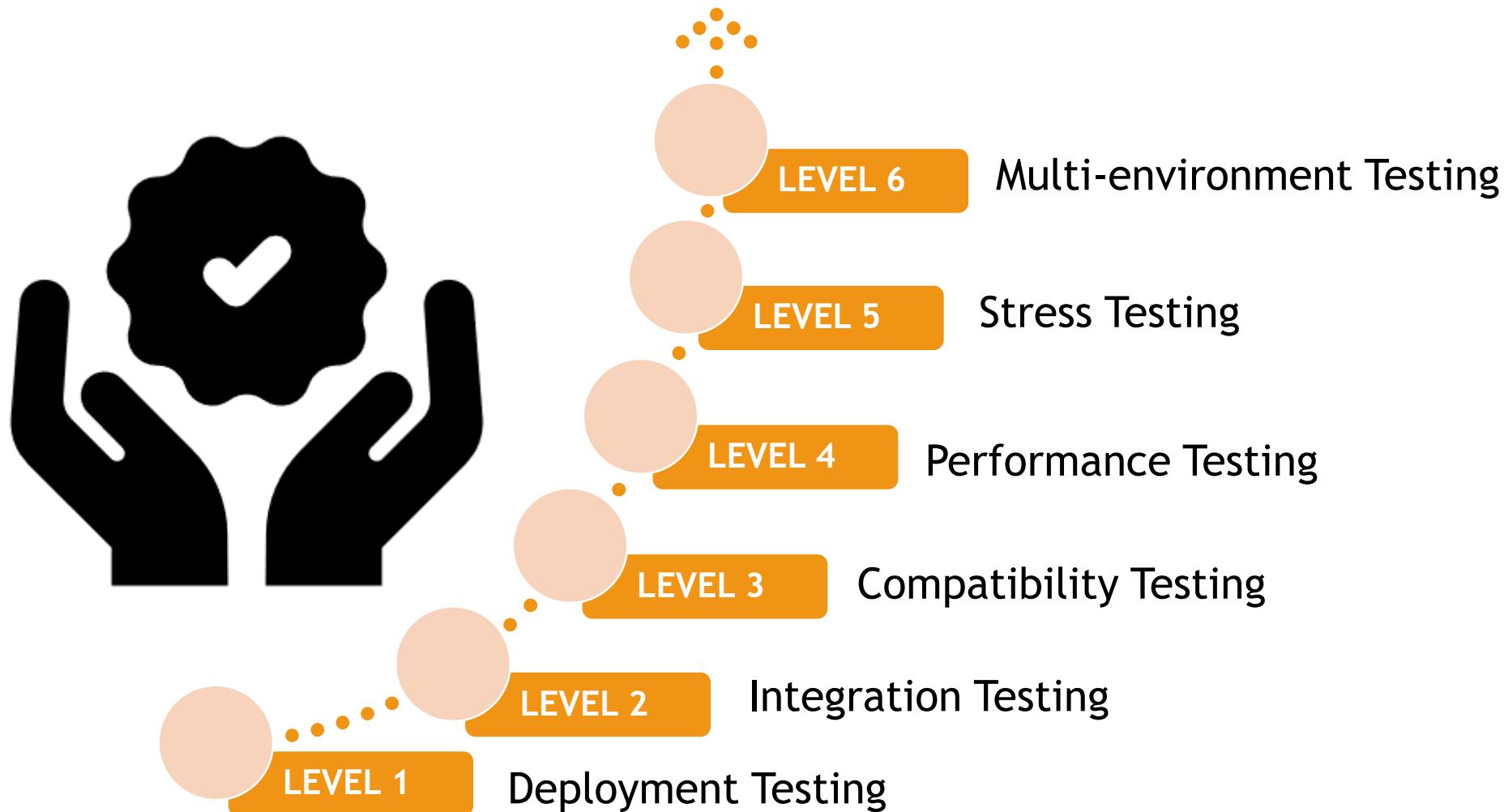
Baseline Design Summary (continued)

Project Elements	Selected Element	Rationale
Packet Analyzer	Wireshark	Lightweight and user-friendly GUI Available for all flavors of OS Supports many protocols Allows the capture on multiple interfaces at once
Database Manager	SQLite	Support for multiple connections Lightweight
Traffic Generator	Iperf	Easy to install and implement across multi-vendor platforms Supports both IPv4 and IPv6 traffic The results are saved in easily readable log files Supports multithreading

Baseline Design Summary (continued)

Project Elements	Selected Element	Rationale
Southbound Protocol	OpenFlow	Standard SDN Protocol Supported by most existing hardware Supports TCP & TLS connection
Northbound Protocol	REST API	Supported by most devices Easy to implement using curl & requests.post method Popular in industry

Verification and Validation



Level 1: Deployment of Network Infrastructure

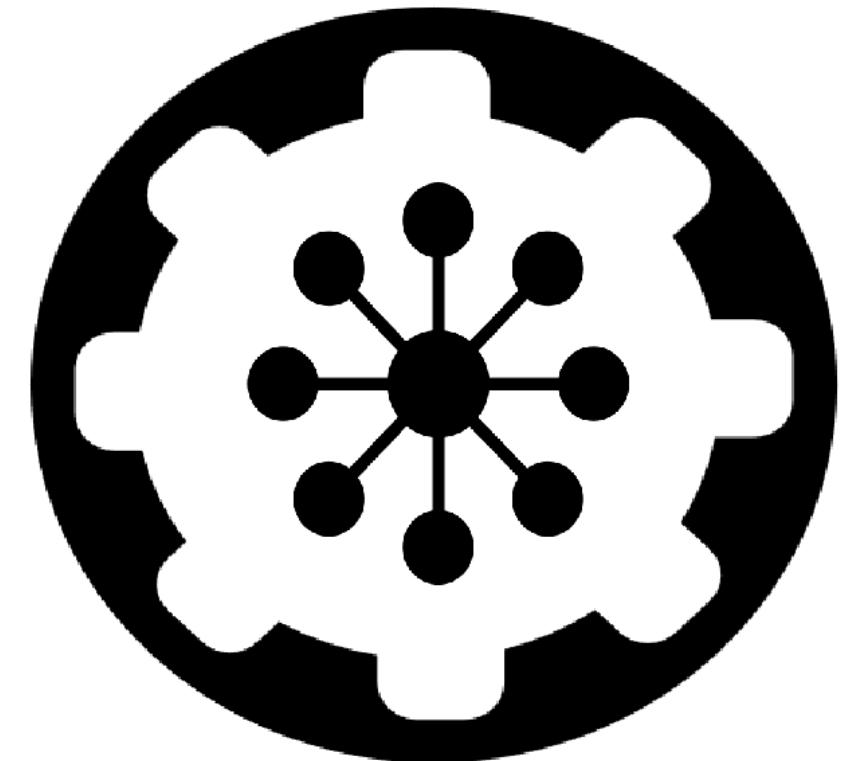
Test the deployment of network devices & software tools

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 Controller VMs using containers. Verify installation of SDN Controller application in the Controller VMs



Level 1: Deployment of Network Infrastructure (continued)

Storage & Output

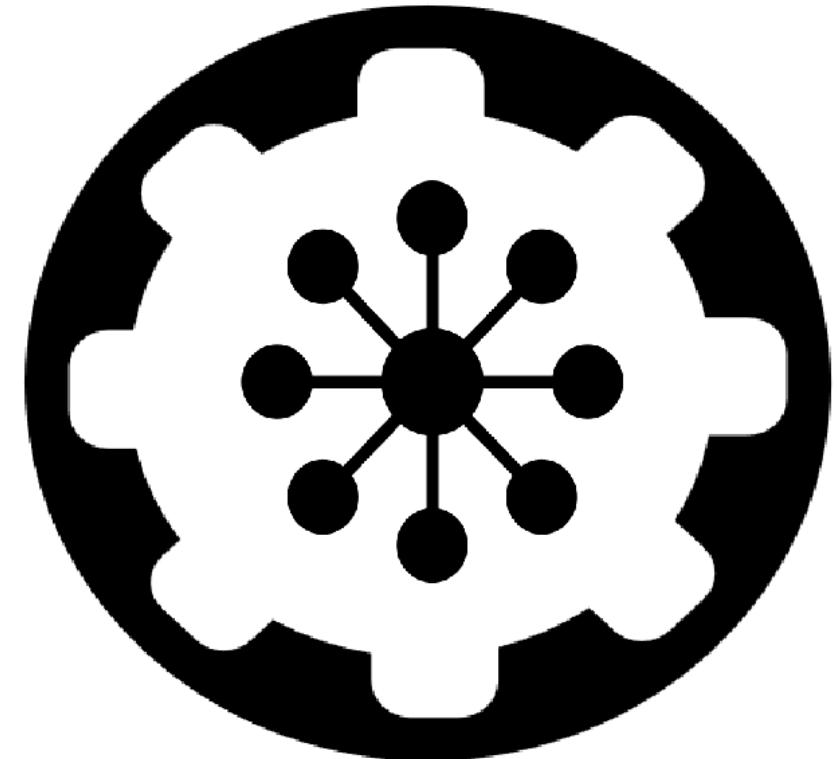
- Validate deployment of storage instances and web interface

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

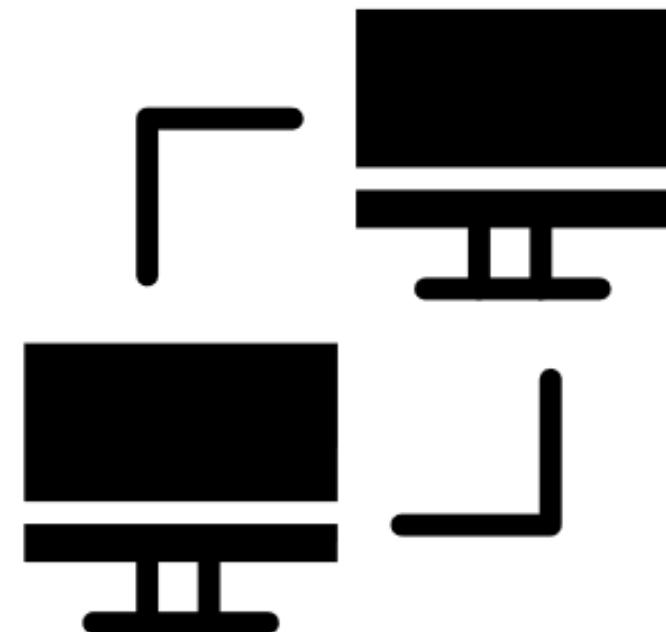
Test the reachability between network devices & VNFs

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

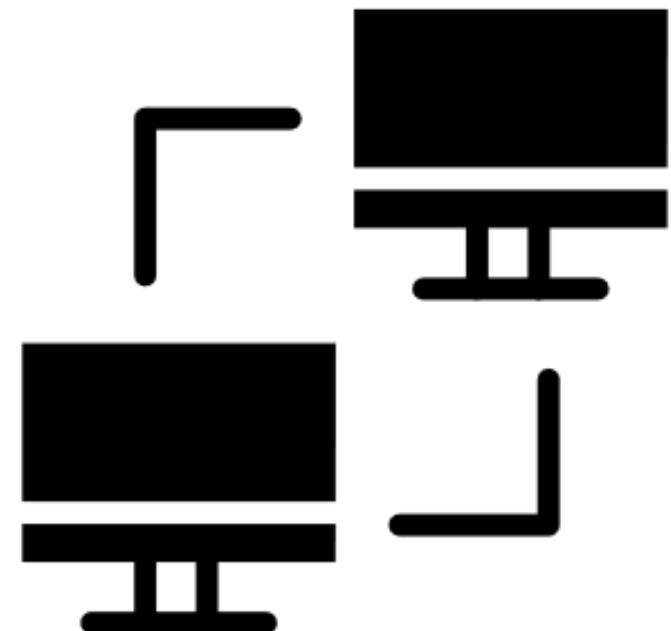
- Validate the connectivity between storage devices and the web interfaces

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

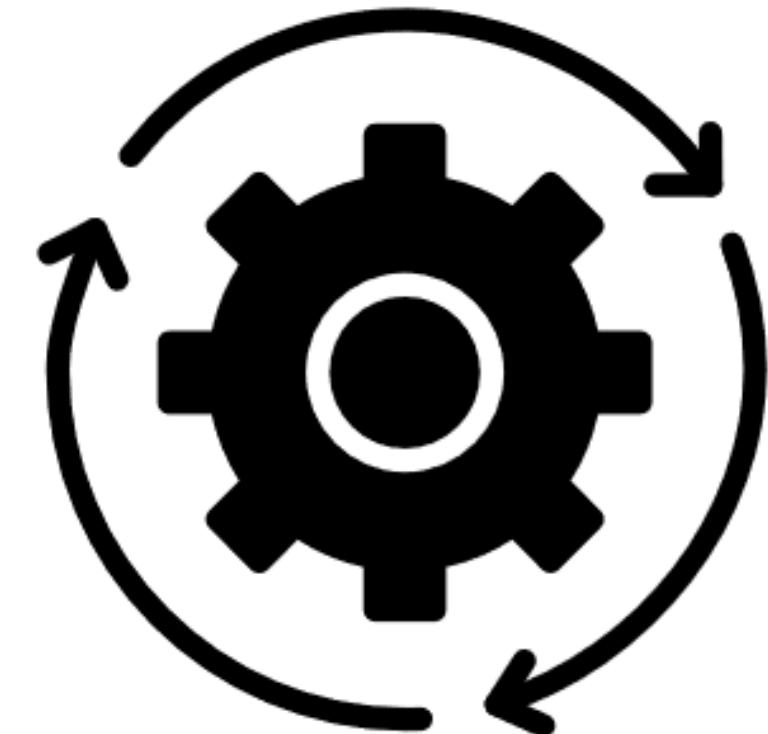
- Verify installation of flow entries in multiple flow tables in the OVS switches

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, jitters, utilization
- Multiple traffic scenarios generated through traffic generator will be used for testing



Level 5: Stress Test for Network Devices

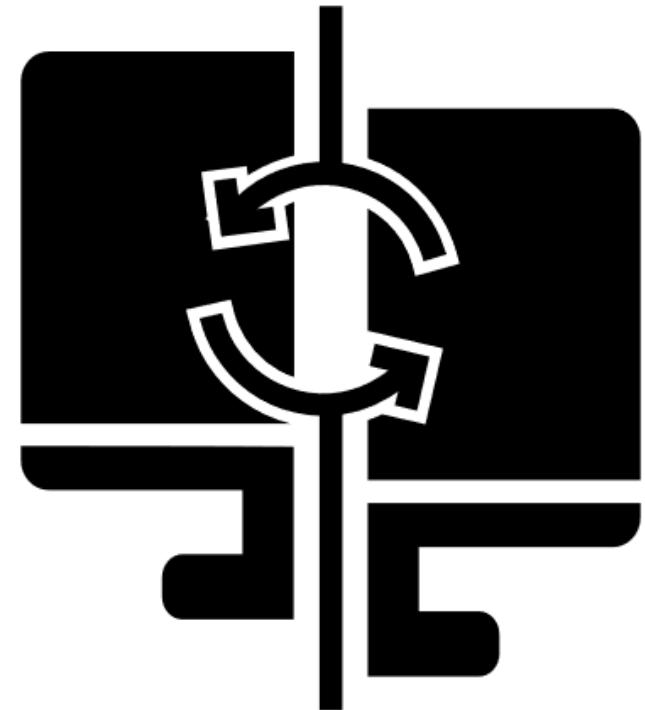
Test the backup scenarios in case of failovers

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 Access point link in case the switch or links goes down



Level 6: Multi-environment Network Deployment

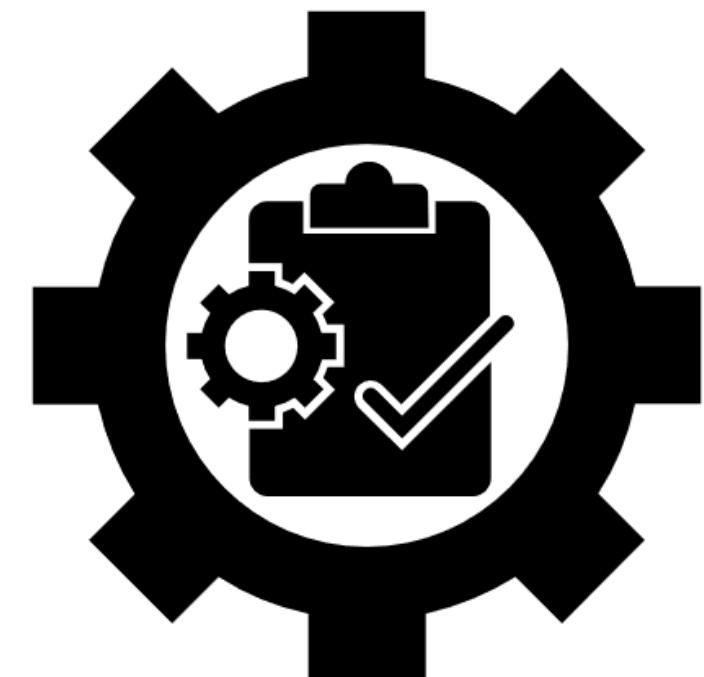
Test the deployment of network services and functionality on different Operating Systems

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



Validation and Verification for Docker

1. Docker installation

```
MINGW64 /c/Program Files/Docker Toolbox
Creating CA: C:\Users\manis\.docker\machine\certs\ca.pem
Creating client certificate: C:\Users\manis\.docker\machine\certs\cert.pem
Running pre-create checks...
(default) Image cache directory does not exist, creating it at C:\Users\manis\.docker\machine\cache...
(default) No default Boot2Docker ISO found locally, downloading the latest release...
(default) Latest release for github.com/boot2docker/boot2docker is v19.03.5
(default) Downloading https://github.com/boot2docker/boot2docker/releases/download/v19.03.5/boot2docker.iso...
  (default) 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Creating machine...
(default) Copying C:\Users\manis\.docker\machine\cache\boot2docker.iso to C:\Users\manis\.docker\machine\machines\default\boot2docker.iso...
(default) Creating VirtualBox VM...
(default) Creating SSH key...
(default) Starting the VM...
(default) Check network to re-create if needed...
(default) Waiting for the host-only adapter for the permission to create a network adapter. Sometimes, such confirmation window is minimized in the taskbar.
(default) Bind a new host-only adapter: "VirtualBox Host-Only Ethernet Adapter #3"
(default) Windows might ask for the permission to configure a network adapter. Sometimes, such confirmation window is minimized in the taskbar.
(default) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation window is minimized in the taskbar.
(default) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Provisioning the instance...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env default
```

3. Docker version

```
manis@LAPTOP-V72I25C2 MINGW64 /c/Program Files/Docker Toolbox
$ docker --version
Docker version 19.03.1, build 74b1e89e8a
```

2. Docker CLI

4. Docker images

```
manis@LAPTOP-V72I25C2 MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY           TAG      IMAGE ID      CREATED       SIZE
globocom/openvswitch latest   58215e48af7c  20 months ago  73.4MB
quay.io/netsys/ovsdb-server latest   934be8fa6fb9d  3 years ago   299MB
quay.io/netsys/ovs-vswitchd latest   d408a41f555d  3 years ago   299MB
quay.io/netsys/ovn-northd latest   664bc7701658  3 years ago   299MB
```

Validation and Verification for SDN Controller

1. VM installation

```
(base) engr2-Z-90-165-dhcp:manesh dhanendrasoni$ nano Dockerfile
(base) engr2-Z-90-165-dhcp:manesh dhanendrasoni$ docker build -t sdn_controller:0.1 .
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM ubuntu:18.04
--> 775349758637
Step 2/2 : RUN apt-get update -y && apt-get install -y python3 python3-pip git && pip3 install pillow jsonpickle && pip3 install ryu
--> Running in 17a39fc84cc2
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [19.2 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [6783 B]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [760 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [794 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [10.5 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [32.7 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1322 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1056 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [4244 B]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [2496 B]
[...]
```

2. VM configuration

```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "957efef1e423d2b1370d746ef936329eeda88633d47a31fddc9947f5d9dd202225",
        "EndpointID": "aa6e0f4bbd093b261813f4806dcdd959655db0512278565bf152412c66d5e56e",
        "Gateway": "172.17.0.1",
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:02",
        "DriverOpts": null
    }
}
```

3. VM image running status

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sdn_controller	0.1	6fe5d6632228	8 seconds ago	607MB
<none>	<none>	245c5ea70a30	3 weeks ago	94.5MB
rest	0.2	d3f49ee6c829	3 weeks ago	598MB

4. SDN Controller application installation

```
root@024c32c8abb1:/# git clone git://github.com/osrg/ryu.git
Cloning into 'ryu'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 26676 (delta 8), reused 8 (delta 3), pack-reused 26653
Receiving objects: 100% (26676/26676), 26.34 MiB / 842.00 KiB/s, done.
Resolving deltas: 100% (19168/19168), done.
root@024c32c8abb1:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run ryu sbin srv sys tmp usr var
root@024c32c8abb1:/# cd ryu/
```

Validation and Verification for OpenvSwitch

1. OVS installation

```
(base) engr2-2-90-165-dhcp:manesh dhanendrasoni$ docker run -i -t --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: Downloading [=====] 7.669MB/23.96MB
d477371faa9b: Download complete
1ff1c88b1fca: Download complete
b04e3c06a506: Download complete
24a45206440d: Download complete
6bd0e8a2022e: Download complete
31354961aae9: Download complete
```

2. OVS version

```
(base) engr2-2-90-165-dhcp:manesh dhanendrasoni$ docker exec -it ba65efb65cb6 /bin/sh
/ # ovs-vsctl -V
ovs-vsctl (Open vSwitch) 2.8.1
DB Schema 7.15.0
/ #
```

3. OVS images- multiple vendors

\$ docker ps	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
	NAMES				
	c08dac5c8af9	quay.io/netsys/ovn-northd	"ovn-northd --log-fi..."	10 minutes ago	Up 10 minutes
	2f8bbe766887	ovn-northd	"ovs-vswitchd --pidf..."	13 minutes ago	Up 13 minutes
	2eda63755692	ovs-vswitchd	"ovsdb-server --remo..."	13 minutes ago	Up 13 minutes
	03ab2178b5c5	ovsdb-server	"ovsdb-server --remo..."	41 minutes ago	Up 41 minutes
	p	58215e48af7c	/bin/sh -c /usr/bin..."	About an hour ago	Up About an hour
	2ba8b4adad4b	gallant_tharp	/bin/sh -c /usr/bin..."	About an hour ago	Up About an hour
	p	58215e48af7c	/bin/sh -c /usr/bin..."	About an hour ago	Up About an hour
	f73f37f7719e	determined_feynman	/bin/sh -c /usr/bin..."	About an hour ago	Up About an hour
		globocom/openvswitch	/bin/sh -c /usr/bin..."	About an hour ago	Up About an hour

4. OVS bridge and controller setup

```
/ # ovs-vsctl add-br br-test
/ #
$ docker exec -it 2eda63755692 /bin/sh
# ovs-vsctl add-br br0
/ # ovs-vsctl set-controller br-test tcp:172.17.0.2:6633
/ # ovs-vsctl set-controller br-test tcp:172.17.0.2:6633
```

5. OpenFlow status

```
$ docker exec -it 2eda63755692 /bin/sh
# ovs-vsctl show
7f0a7a09-bba0-4058-9685-83f5fcfd9aa13
  Bridge "br0"
    Port "br0"
      Interface "br0"
        type: internal
  Bridge br-test
    Port br-test
      Interface br-test
```

Validation and Verification: Integration & Compatibility

1. Reachability between controller and OVS

```
/ # ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.100 ms
64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.105 ms
64 bytes from 172.17.0.3: seq=2 ttl=64 time=0.113 ms
64 bytes from 172.17.0.3: seq=3 ttl=64 time=0.112 ms
^C
--- 172.17.0.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.100/0.107/0.113 ms
```

2. SDN controller initiation

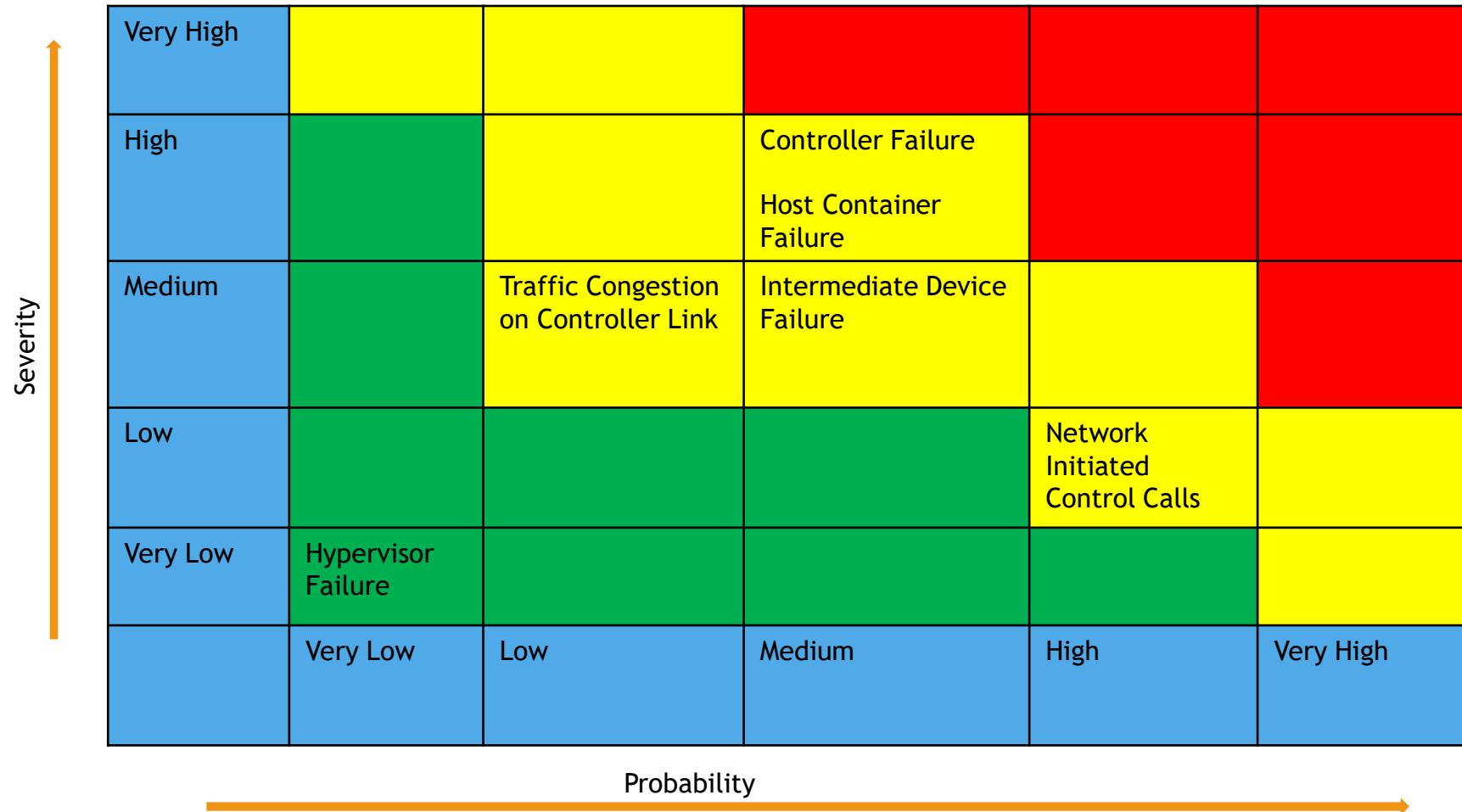
```
root@17ef16693b67:/# cd ryu/ryu/app/
root@17ef16693b67:/ryu/ryu/app# ryu-manager simple_switch.py
loading app simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

3. Southbound protocol connectivity

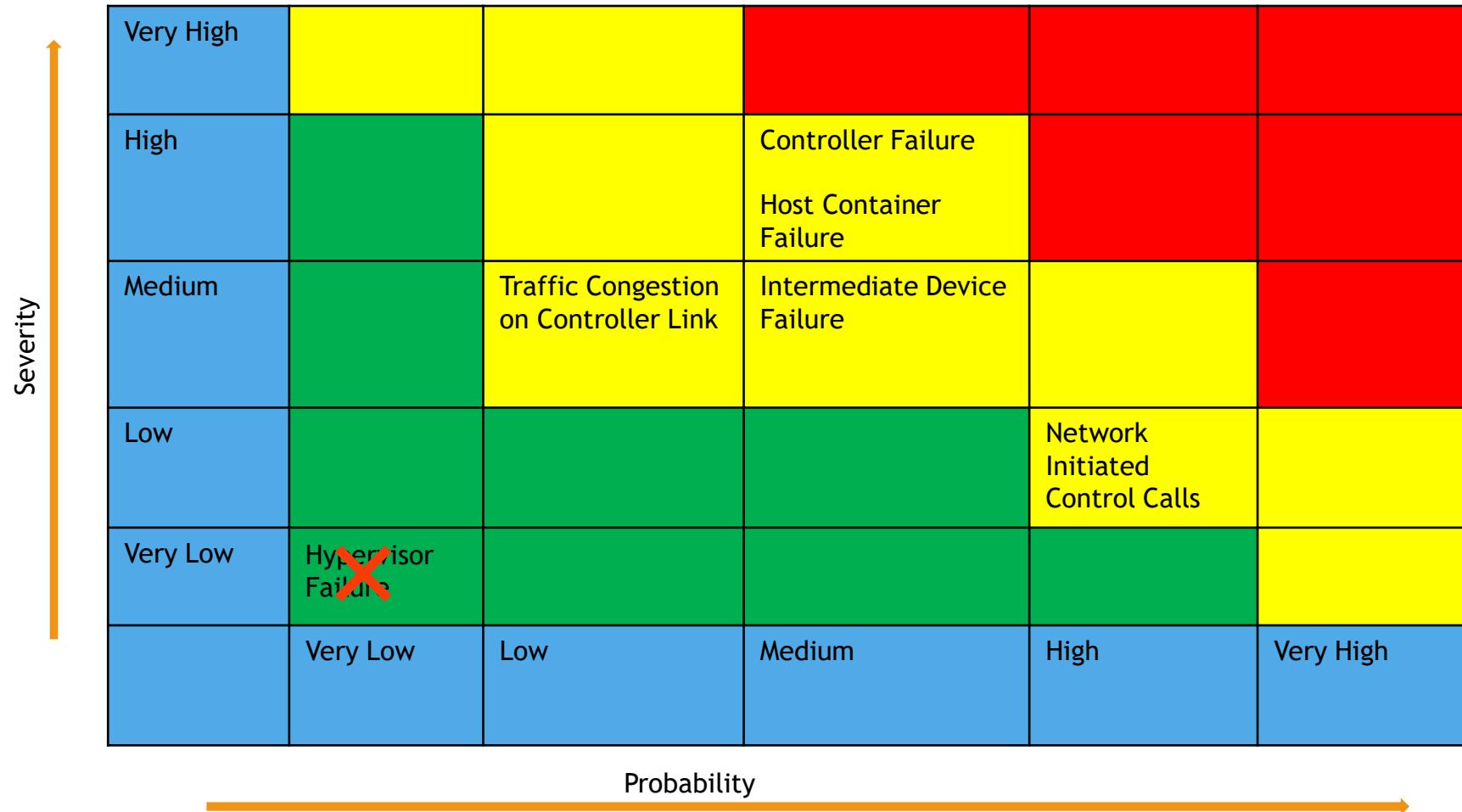
```
loading app simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler

packet in 138910711198538 00:23:20:a4:81:5a ff:ff:ff:ff:ff 65534
```

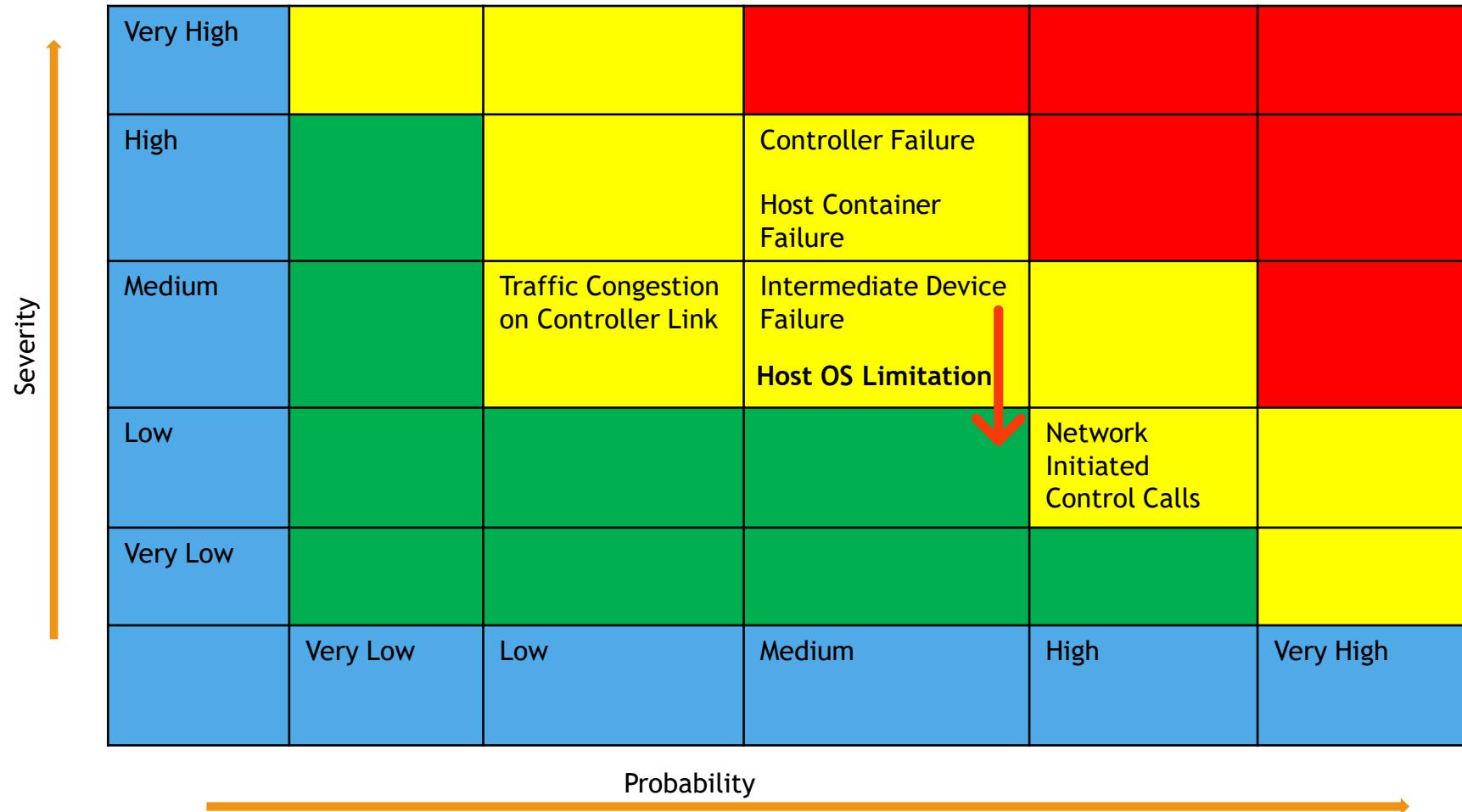
Risk Factors and their Impact - PDR



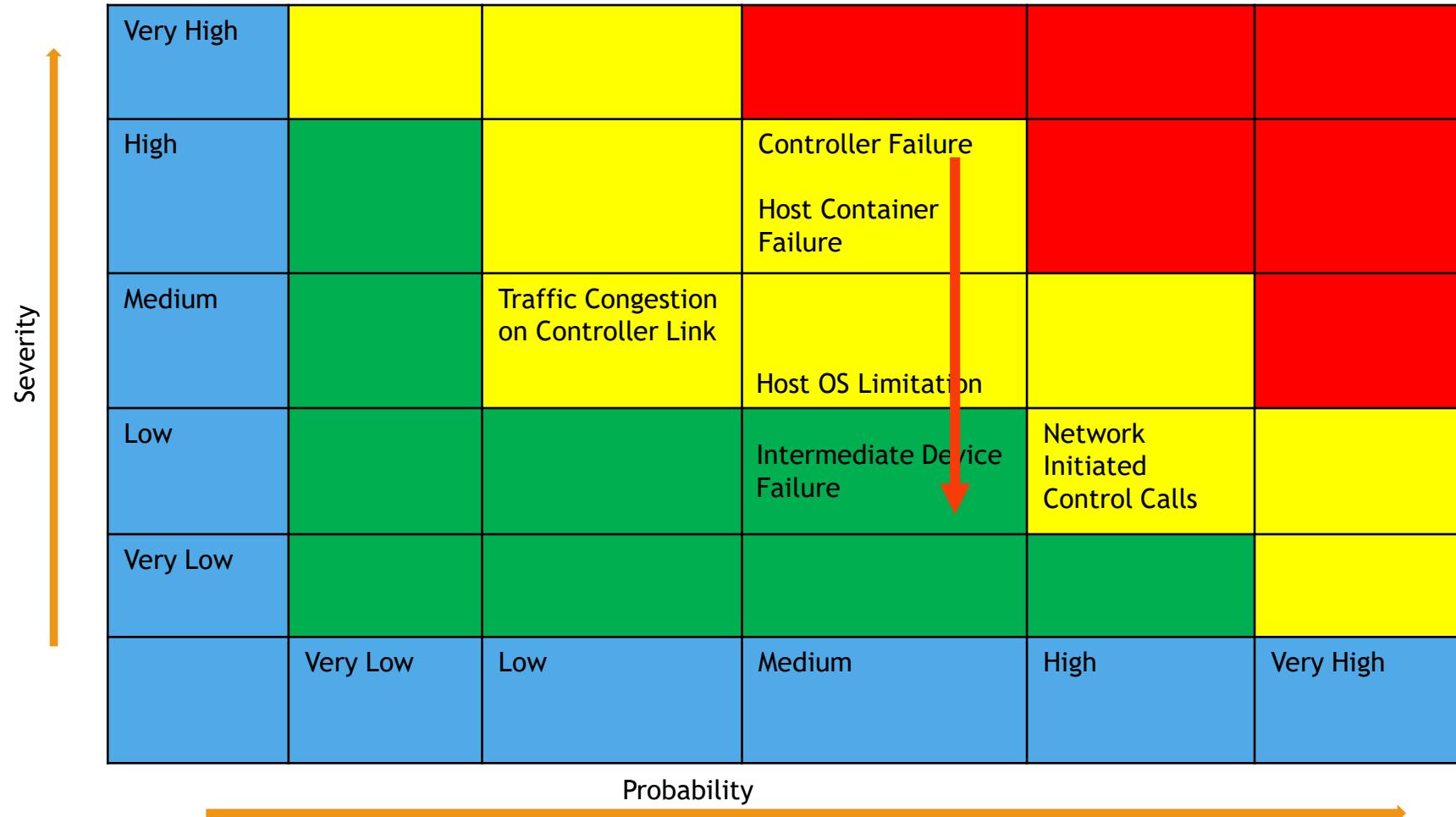
Risk Factors and their Impact - CDR



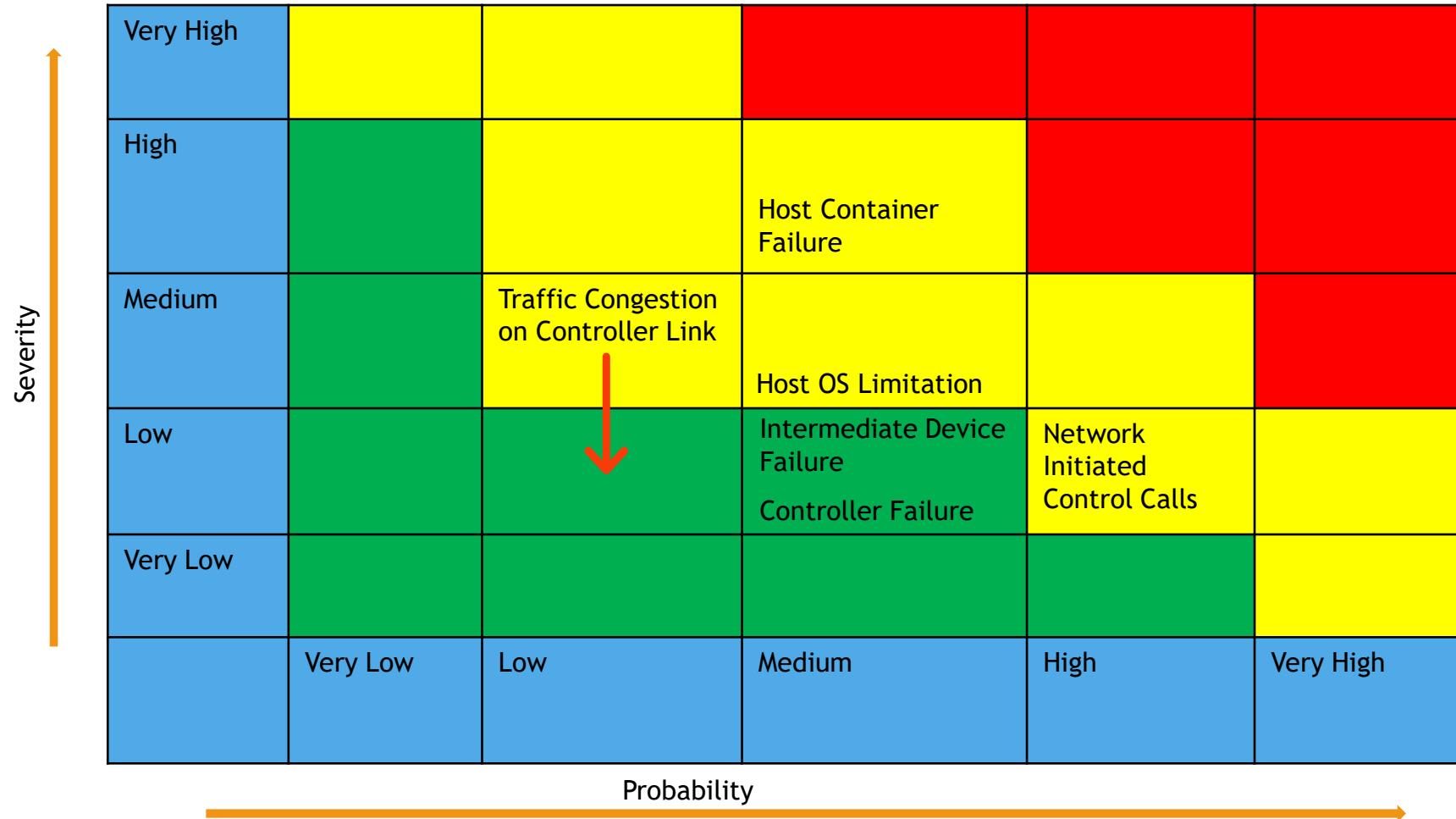
Risk Factors and their Impact - CDR



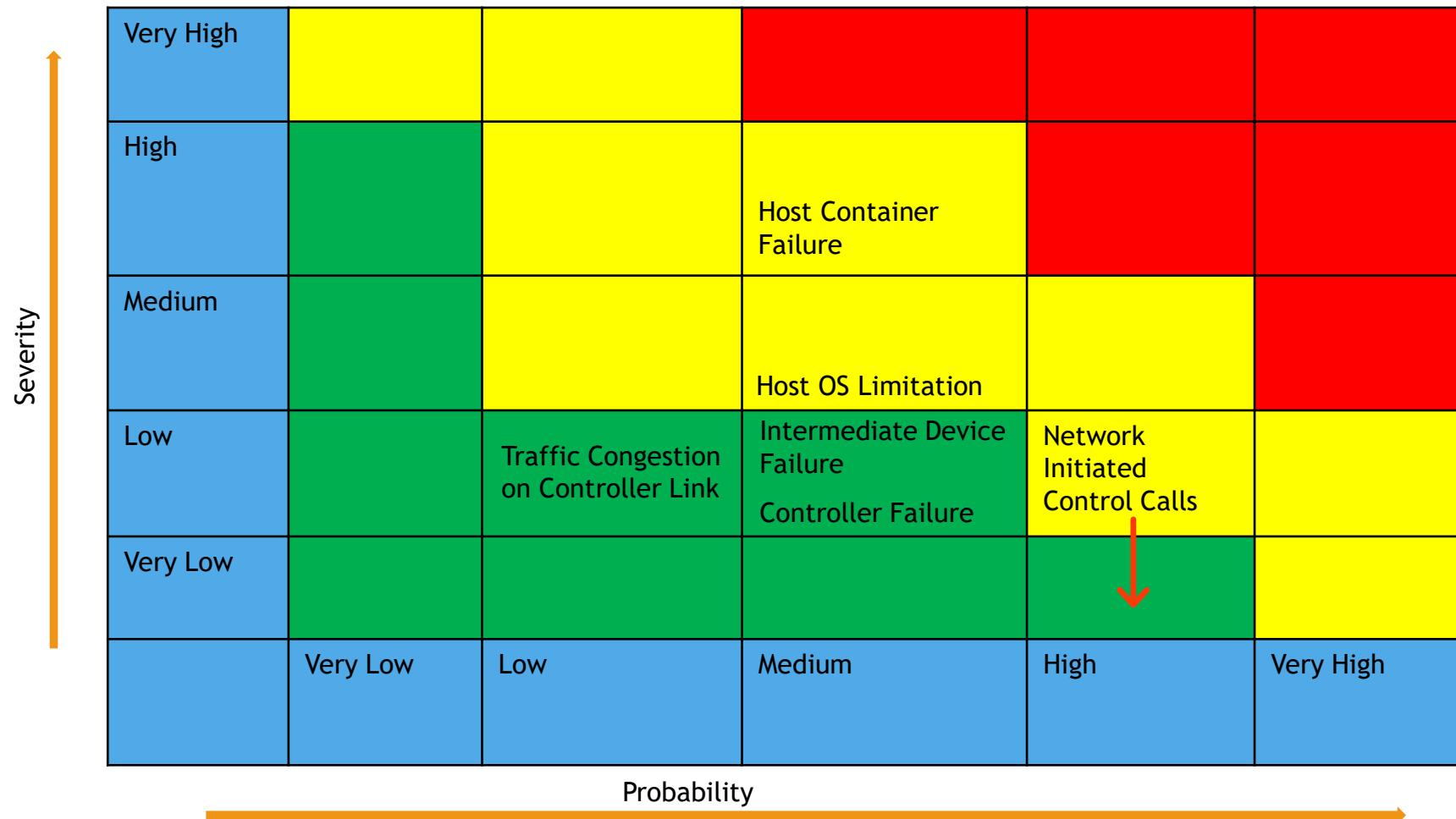
Risk Factors and their Impact - CDR



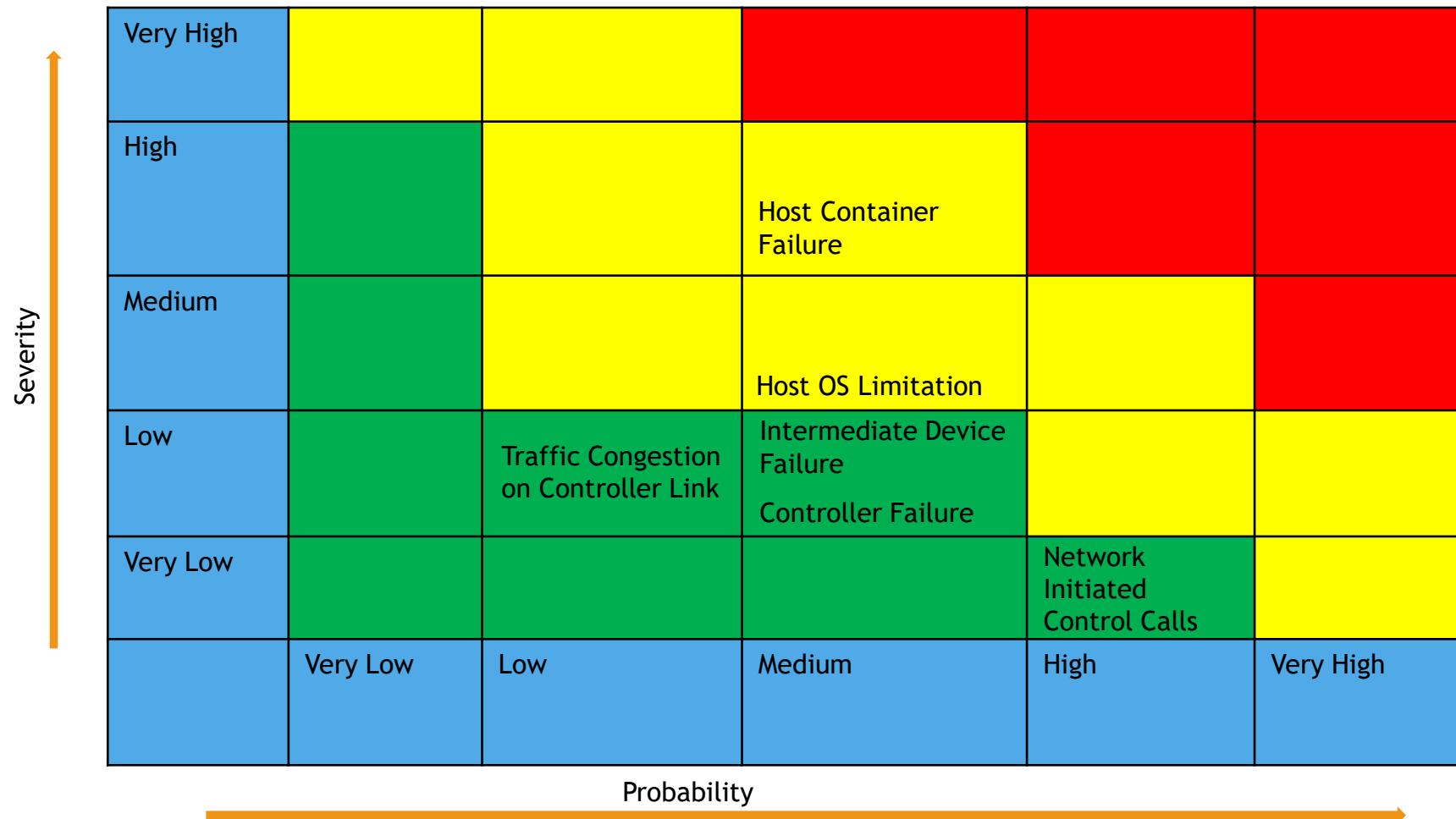
Risk Factors and their Impact - CDR



Risk Factors and their Impact - CDR



Risk Factors and their Impact - CDR



Project Risks: Hardware Failures

Controller Failure

Solution
Introduce a Backup Controller

- Future Work**
- Deploy a backup server
 - Install ONOS on both main and backup servers in a distributed setup

Host Container Failure

Solution
Spin up a backup container on test VM

- Future Work**
- Deploy a backup server with high compute and storage to support backup container

BACKUP CONTROLLER AND TEST VM HARWARE



Product: DELL POWEREDGE R430 1U RACK SERVER
Features:

- Processor: Intel® Xeon® processor E5-2600 v4
- Memory: 64GB
- Storage: 500GB SATA, 146GB SAS
- Networking: 4-1GB Ethernet Network Interface Card

Project Risks: Hardware Failures (continued)

Intermediate Device Failure

Solution

- Use a wireless router with L2 capabilities for redundant connection between devices

Future Work

- Test and deploy Cisco wireless router in the Telecommunications LAB to support high speed wired and wireless connection between all devices

BACKUP HARWARE FOR REDUNDANT CONNECTIONS



Product: CISCO 1811W Wireless Router
Features:

- Connectivity: Wired and Wireless
- Wireless Protocol: 802.11a/b/g
- Max Range Indoors: 320fts
- Integrated Switch: 8 port Switch

Project Risks: Non-hardware Failures

Host OS Software support(Windows only) for container management

Solution

Obtain an Enterprise Windows OS edition with intel vPro support

Future Work

- Discuss with Prof Jose about obtaining right host laptops with software support
 - Fall back to non-windows hosts if we unable to obtain software support

Traffic Congestion on Controller Link

Solution

Backup secondary link for test traffic

Future Work

- Test and deploy dual connection to controller via two NIC cards for OpenFlow and test bed traffic segregation

Network Control Calls

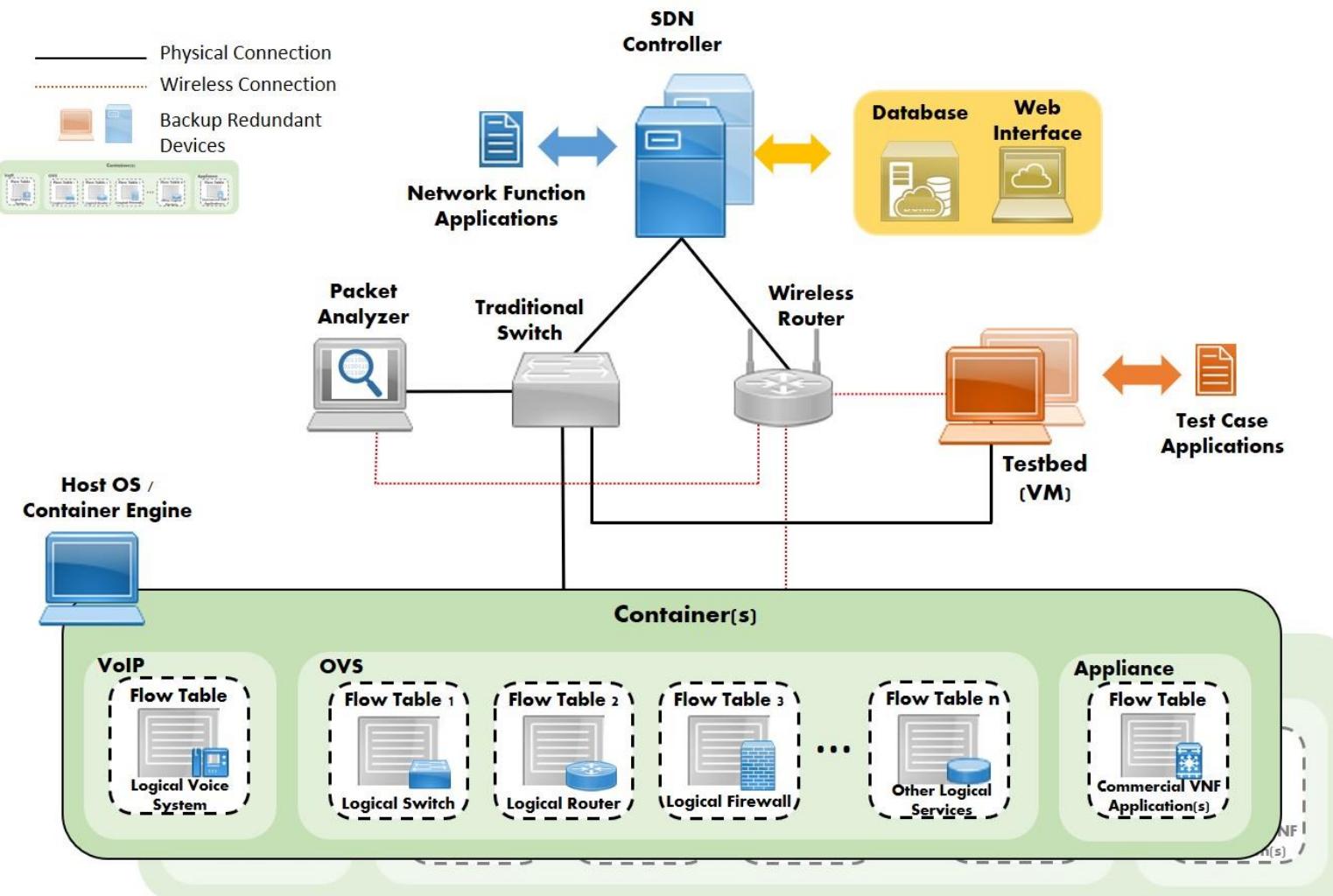
Solution

Implement proactive flows

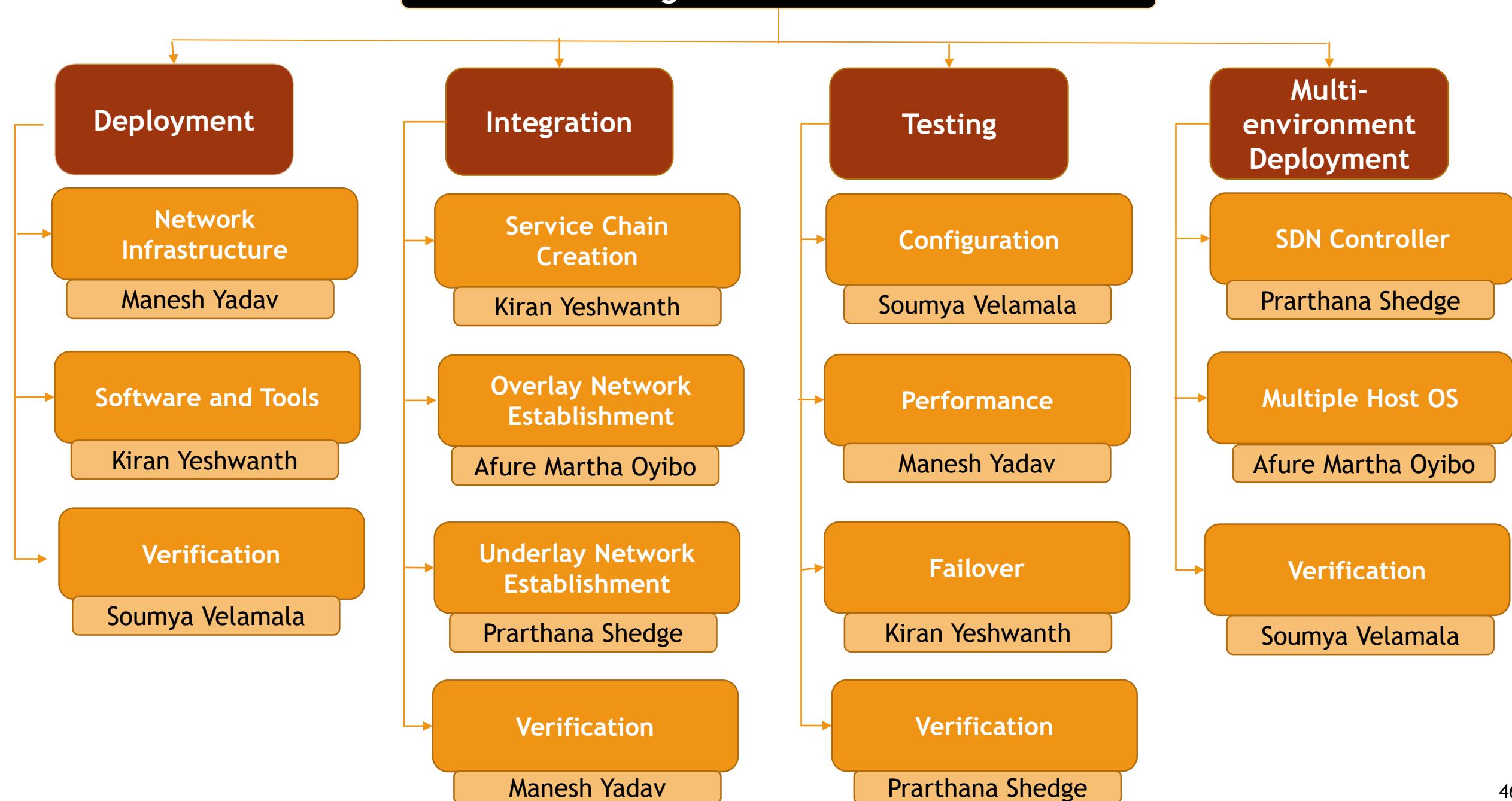
Future Work

- Test multiple proactive flows based on topology to reduce network calls

Revised CONOPS



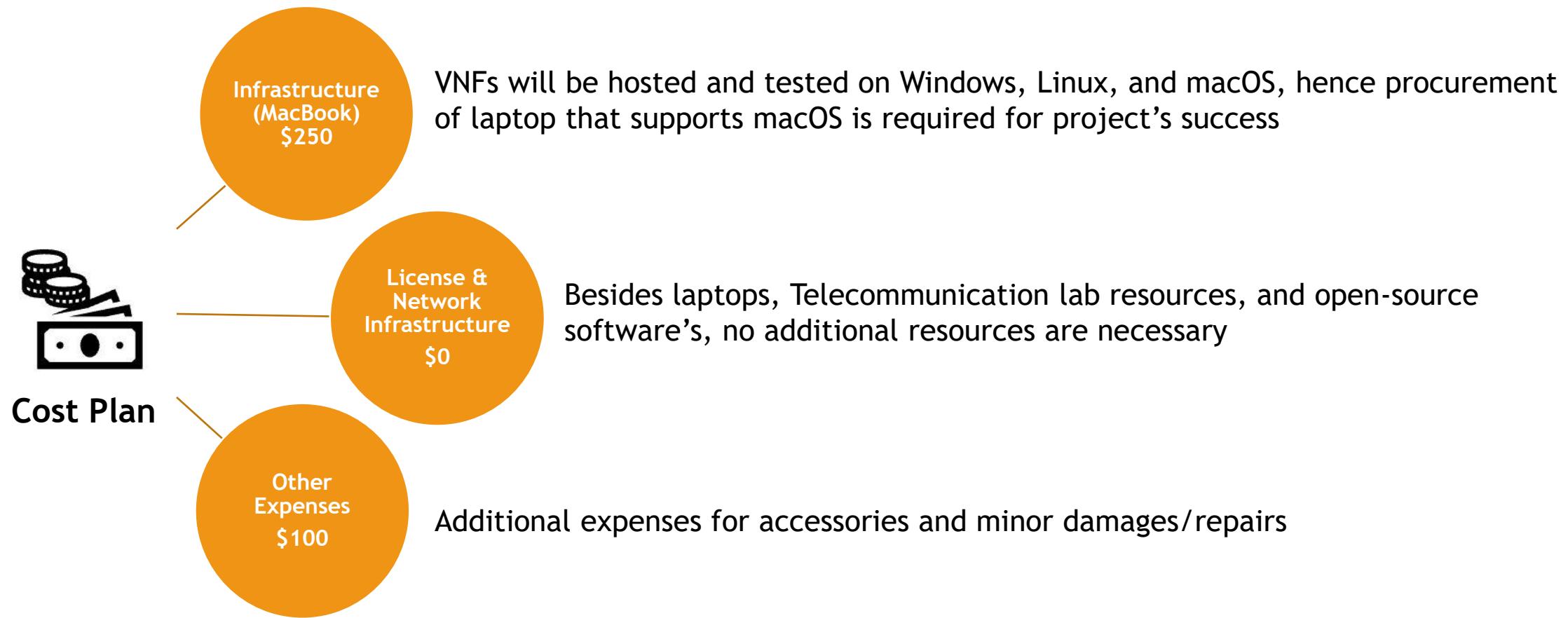
Organizational Chart



Work Breakdown Structure

Network Infrastructure Setup	Deploy VNFs using containers	Phase 1
	Deploy SDN Controllers using containers	
	Create test VM on hypervisor	
	Setup traditional switches to form the backbone network	
Software and Tools Installation	Web GUI creation	Phase 2
	Database management system installation	
	Setup traffic generator and analyzer	
	Service chain creation between VNFs	
	Network orchestration	
	Establish underlay and overlay connectivity	
	Integrate web interface with database management system	
	Integrate test environment on test VM	
	Establish connectivity between test VM and database management system	
	Establish Northbound communication between Controller and Application layer	
Network Integration	Establish Southbound communication between Controller and VNFs	Phase 2
	Establish end-to-end service accessibility	
	Automate network configuration	
	Test configuration and active status of network infrastructure	
	Test connectivity between topology devices	
Testing and Verification	Test VNFs behavior for different traffic scenarios	Phase 3
	Compare performance of VNFs in comparison to traditional networks	
	Detect network failures and verify activation of backup solutions	
	Enable provisioning and orchestration of network services via web interface	
One-Touch Orchestration		

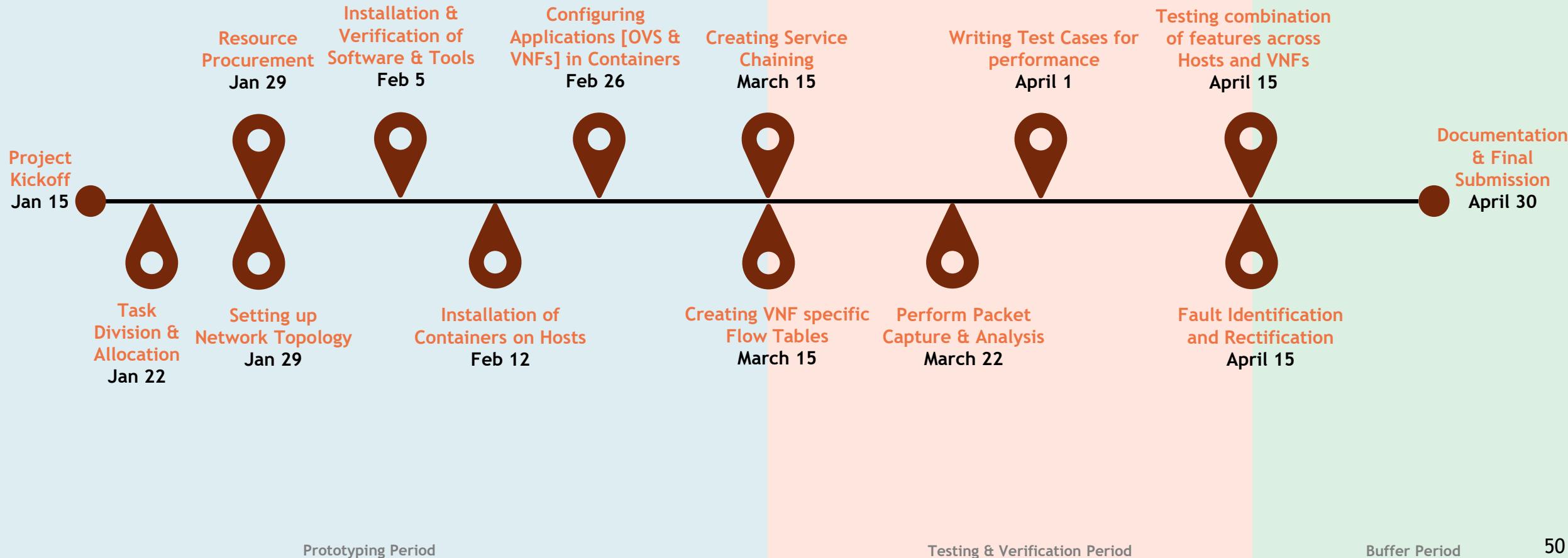
Cost Plan



Hardware Requirements

Hardware Type	Model	Quantity
Server	Dell PowerEdge	2
Switch	Cisco 2960 Cisco 1811W	1
Laptops	Windows/MAC /Linux	3

Project Targets and Key Deadlines



Q&A

Thank You

Backup Slides

Evaluation Parameters



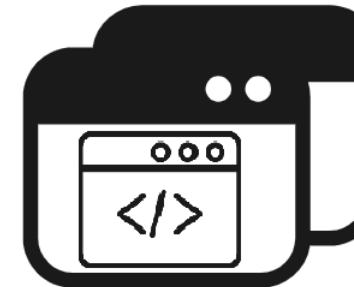
Cost



Open-source



Features



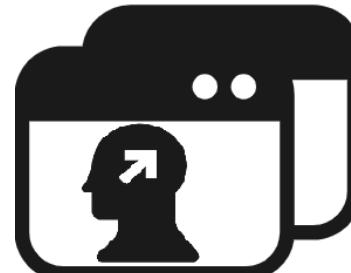
Programmability



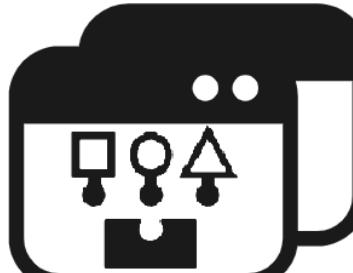
Community Support



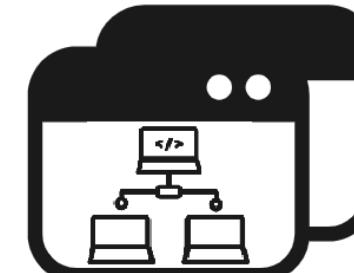
Documentation



Learning Curve



Modularity



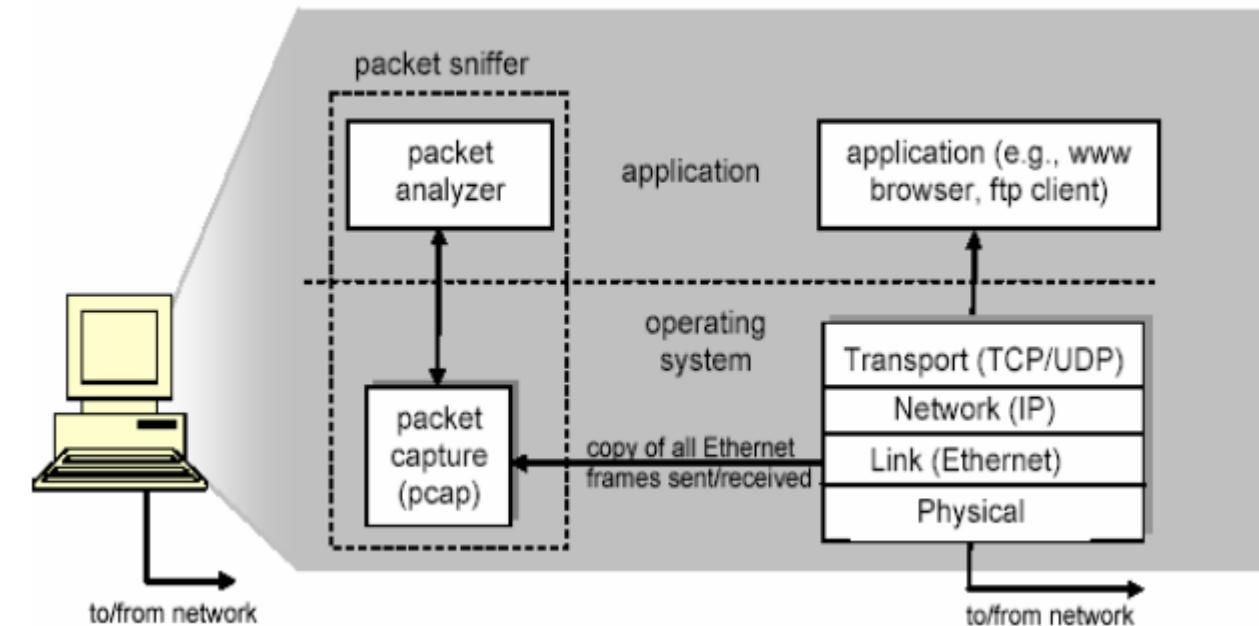
Ease of Deployment



Bare Metal Speed

Packet Analyzer

- Software that is used to capture and perform packet analysis of the network traffic
- Visualization of network bandwidth and resource utilization [5]



Baseline Design from CDD - Packet Analyzer

Comparison of Packet Analyzers

Parameters	Weights	Wireshark	Tshark	Tcpdump
Cost	0.20	5	5	5
Open-source	0.15	5	5	5
Feature Support	0.10	4	4	3
Programmability	0.15	5	5	4
Community Support & Documentation	0.15	5	4	4
Learning Curve	0.10	4	4	3
Modularity	0.15	5	5	4
Total	1.00	4.80	4.65	4.15

5 - Best choice for the Project

1 - Worst choice for the Project

Pros & Cons of Wireshark Packet Analyzer

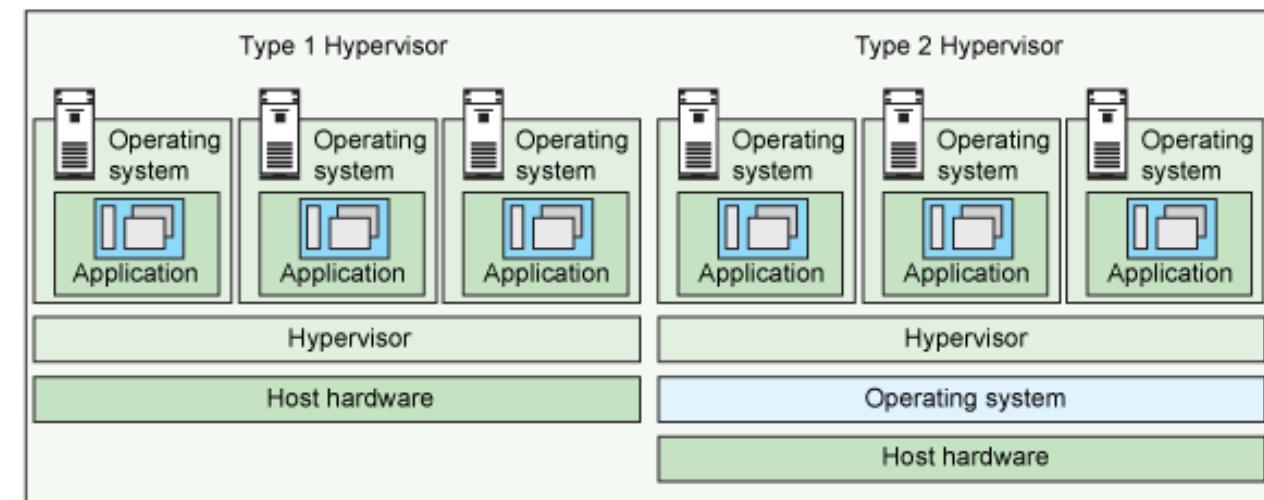
Lightweight and user-friendly GUI
Available for all flavors of OS

Supports many protocols
Allows the capture on multiple interfaces at once

Capture files becomes extremely large, making arduous to identify required packets

Hypervisors

- Software that enables multiple Operating Systems to share the same hardware resources
- Controls the allocation of physical host hardware and kernel resources such as memory, disk space on each virtualized OS



Baseline Design from CDD - Hypervisors

Pros & Cons of KVM Hypervisor

Better support for network virtualization
Free and open-source

Enhanced VM security and isolation
Cheaper to implement

Poor support for large scale storage virtualization

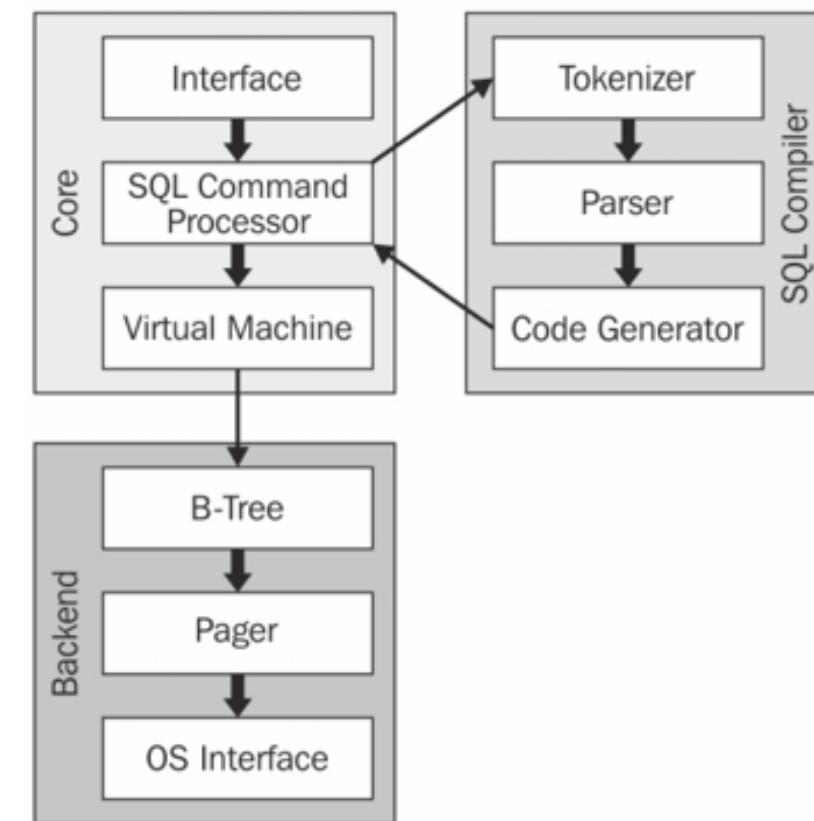
Comparison of Hypervisors

Parameters	Weights	ESXi	KVM	Hyper V	VMware Workstation	VirtualBox	QEMU
Cost	0.20	2	5	2	3	5	5
Open-source	0.15	4	4	2	4	4	4
Feature Support	0.10	3	4	3	4	4	3
Community Support & Documentation	0.15	3	4	3	4	3	3
Learning Curve	0.10	3	4	3	4	4	3
Modularity	0.15	2	4	2	4	4	3
Bare Metal Speed	0.15	4	4	4	2	2	2
Total	1.00	2.95	4.20	2.65	3.50	3.75	3.40

5 - Best choice for the Project
1 - Worst choice for the Project

Database Storage

- Logical Structures: Used to store and index data
- Collection of data can be accessed, updated and analyzed using tools



Baseline Design from CDD - Database Storage

Pros & Cons of SQLite Database Storage

Lightweight application
Limited data type support

Support for multiple connections

Support only one connection at a time
Performance degrades with large set of data
No security feature

Comparison of Database Storages

Parameters	Weights	Oracle	MySQL	SQLite
Cost	0.20	1	5	5
Open-source	0.15	1	5	5
Feature Support	0.10	2	4	3
Programmability	0.15	3	4	4
Community Support & Documentation	0.15	3	4	4
Learning Curve	0.10	2	3	4
Modularity	0.15	2	4	4
Total	1.00	1.95	4.25	4.25

5 - Best choice for the Project
1 - Worst choice for the Project

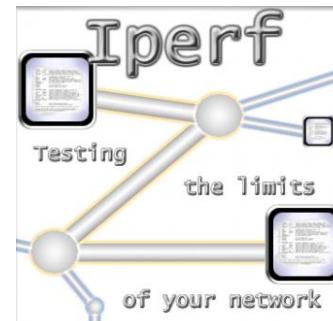
Traffic Generator

- Tool used to generate customized traffic and monitor network performance parameters
- Test the robustness of network under different network scenarios

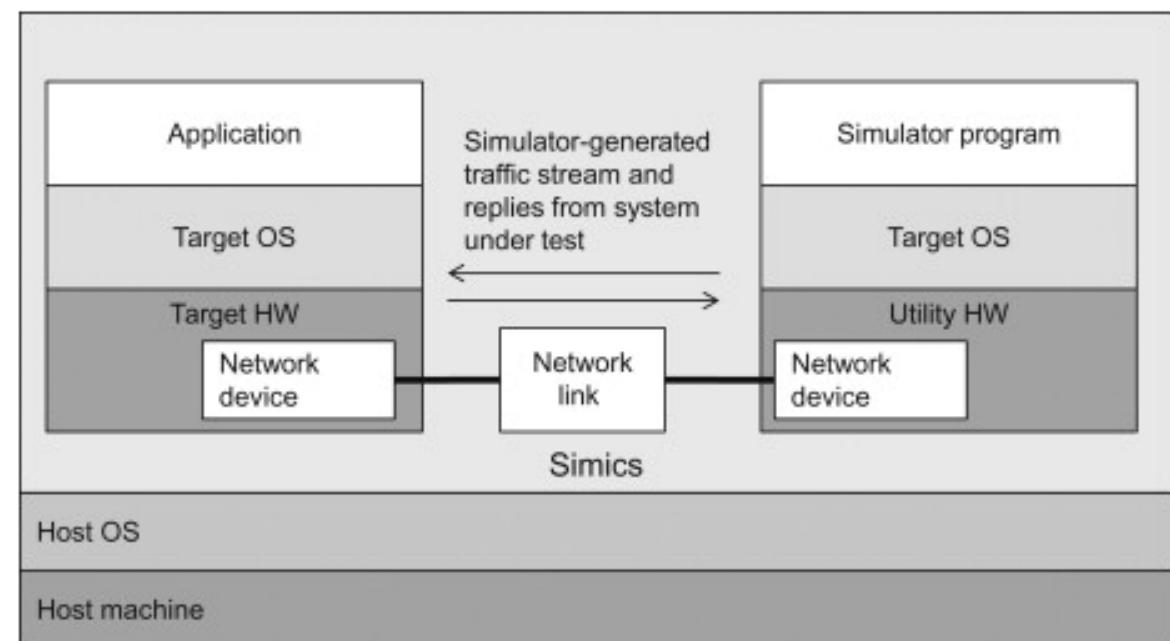


OSTINATO

Network Traffic Generator and Analyzer



íxia



Baseline Design from CDD - Traffic Generator

Comparison of Traffic Generators

Parameters	Weights	Ostinato	Iperf	Ixia	WAN Killer
Cost	0.20	5	5	2	2
Open-source	0.15	4	4	2	2
Feature Support	0.10	2	4	3	2
Programmability	0.15	4	4	3	2
Community Support & Documentation	0.15	2	4	3	4
Learning Curve	0.10	2	4	2	2
Modularity	0.15	3	4	2	2
Total	1.00	3.35	4.20	2.40	2.30

5 - Best choice for the Project

1 - Worst choice for the Project

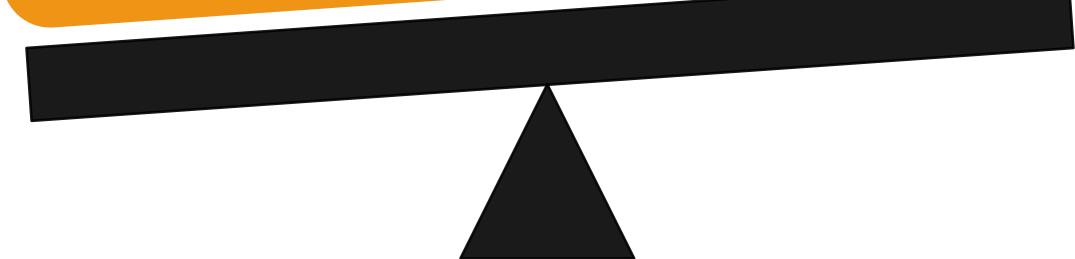
Pros & Cons of Iperf Traffic Generator

Easy to install and implement across multi-vendor platforms
Free of cost

Supports both IPv4 and IPv6 traffic
Huge number of deployments in the industry leading to better support

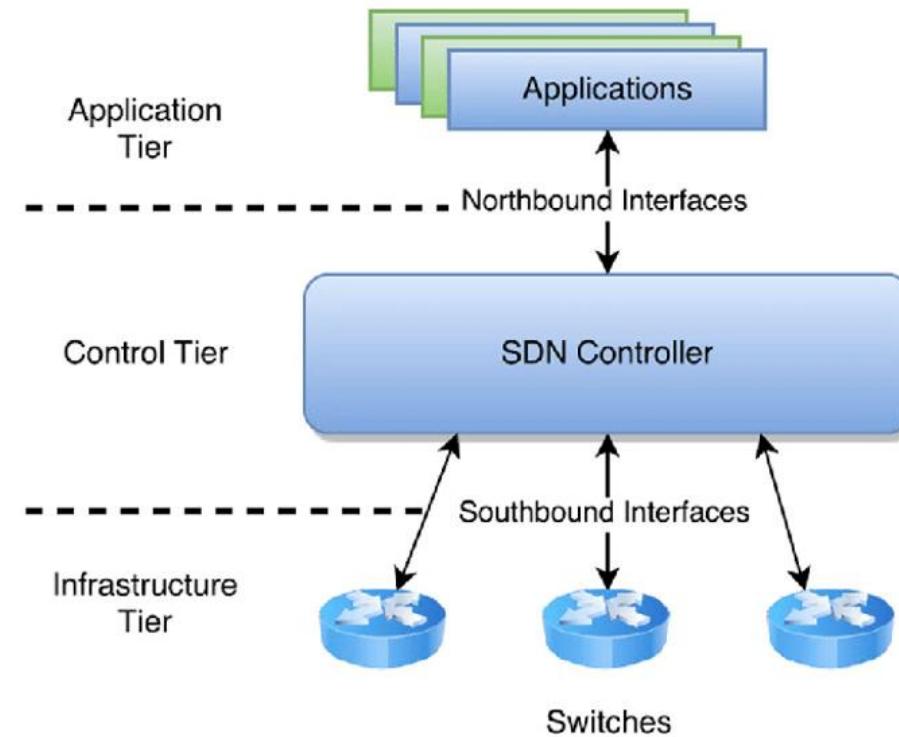
Good documentation
The results are saved in easily readable log files
Supports multithreading

Cannot monitor CPU usage and memory utilization of intermediate nodes
GUI support is not great
With multiple streams, dedicated scripts are necessary to connect client to server
Does not support L7 signature-based traffic generation



SDN Controller

- Application layer in SDN: Responsible for provisioning, orchestration and abstraction of flows in the network devices
- Brain of the network
- Global view
- Open-source vs Commercial



Baseline Design from CDD - SDN Controller

Comparison of SDN Controllers

Parameters	Weights	ONOS	ODL	Ryu	Floodlight	Cisco APIC
Cost	0.20	5	5	5	5	1
Open-source	0.15	4	3	4	4	1
Feature Support	0.10	4	2	3	4	2
Programmability	0.15	5	3	4	2	3
Community Support & Documentation	0.15	4	2	3	3	3
Learning Curve	0.10	4	2	3	3	3
Modularity	0.15	4	4	3	3	2
Total	1.00	4.35	3.20	3.70	3.50	2.05

5 - Best choice for the Project

1 - Worst choice for the Project

Pros & Cons of ONOS SDN Controller

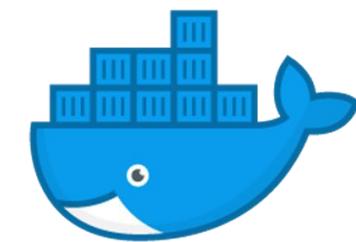
Well documented
High scalability
Interactive GUI framework which can be easily customized as per the requirement
Enhanced security features

Provides northbound abstraction using REST, gRPC or native interface
Regular updates available
Supports multiple southbound protocols such as OpenFlow, P4, Network Configuration Protocol (NETCONF)

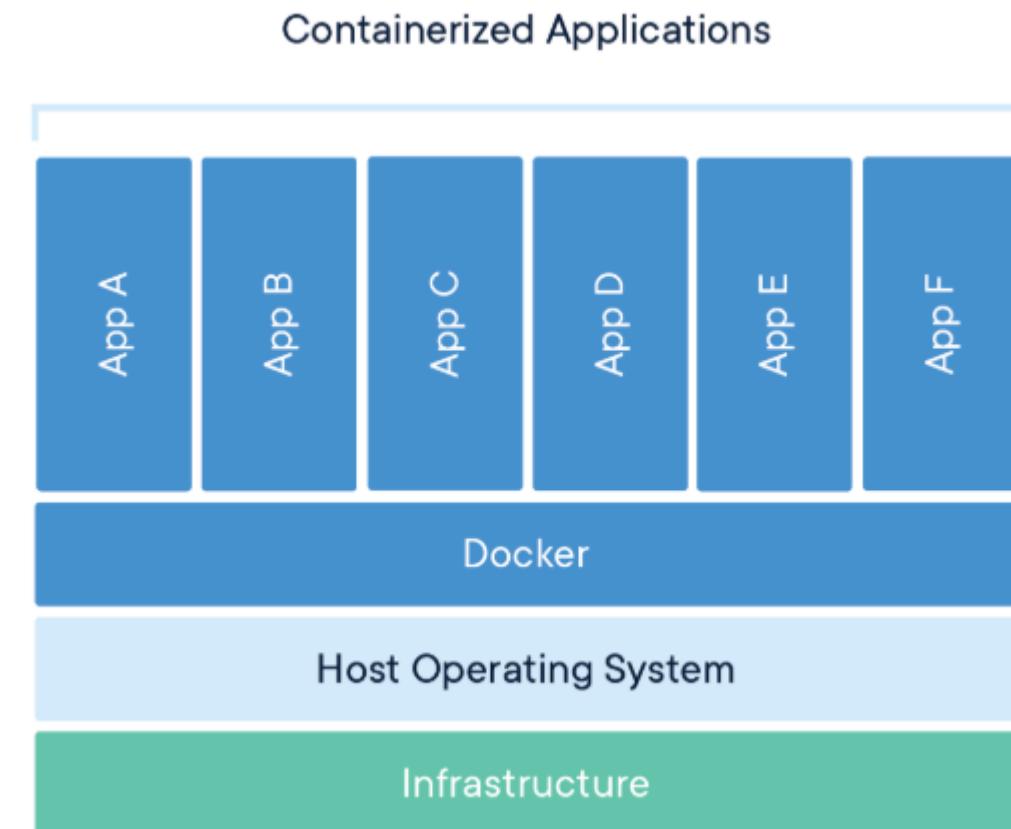
Not suitable for cloud computing and data center architecture
Not compatible with legacy network
Provides sub-optimal performance when installed on Windows OS

Containers

- Standalone, executable software packages: Encases code, user libraries and other dependencies needed to run an application
- Easy to deploy
- Operates on Hosts Operating System

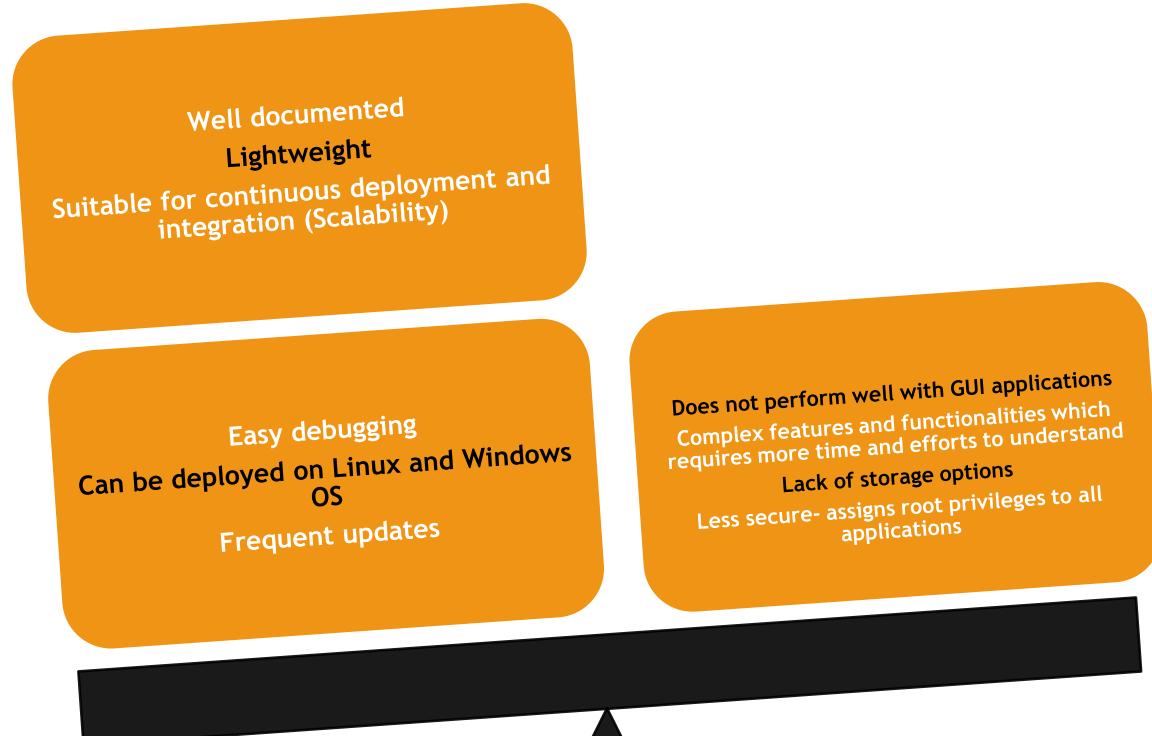


docker



Baseline Design from CDD - Containers

Pros & Cons of Docker Containers



Comparison of Containers

Parameters	Weights	Docker	Rocket	Solaris Containers
Cost	0.20	5	5	1
Open-source	0.15	4	3	1
Compatibility with different OS	0.20	4	2	1
Community Support & Documentation	0.15	4	4	3
Learning Curve	0.10	3	4	4
Ease of Deployment	0.20	4	3	2
Total	1.00	4.10	3.45	1.80

5 - Best choice for the Project
1 - Worst choice for the Project

Validation and Verification for Hypervisor

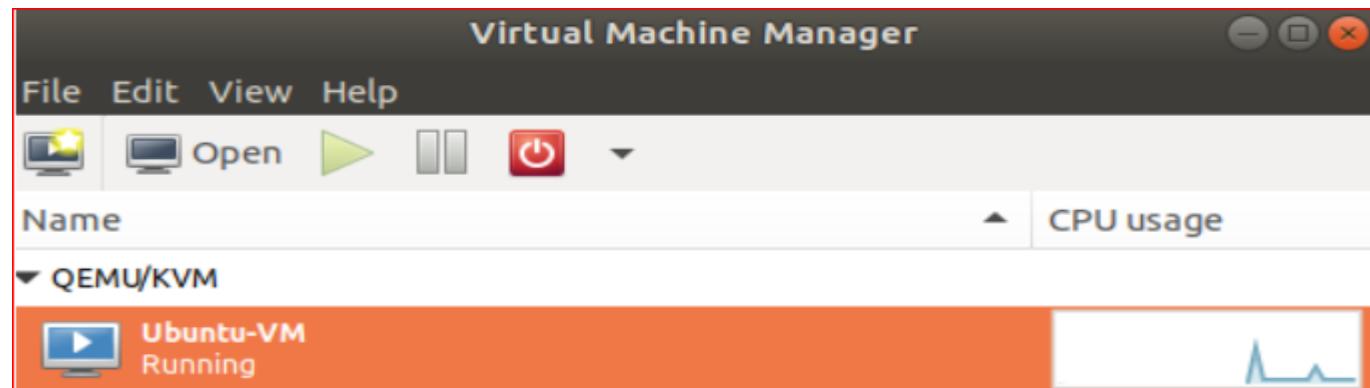
1. Hypervisor virtualization support

```
netman@netman-virtual-machine:~$ kvm-ok  
INFO: /dev/kvm exists  
KVM acceleration can be used
```

2. Verify hypervisor installation

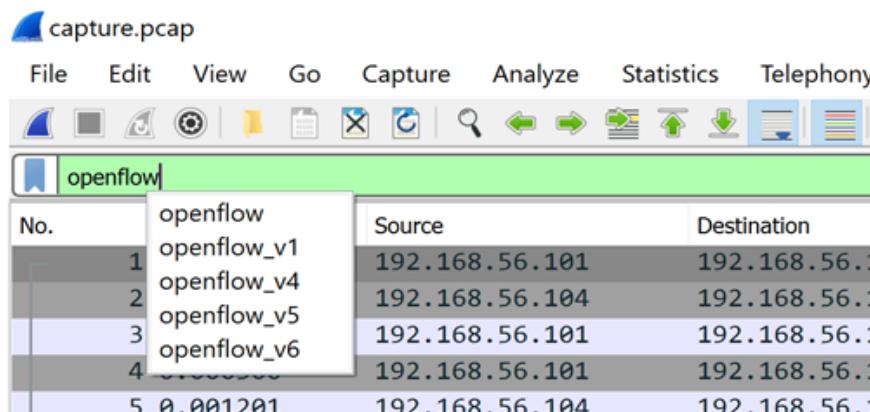
```
netman@netman-virtual-machine:~$ virsh list --all  
Id   Name           State
```

3. Successful VM creation

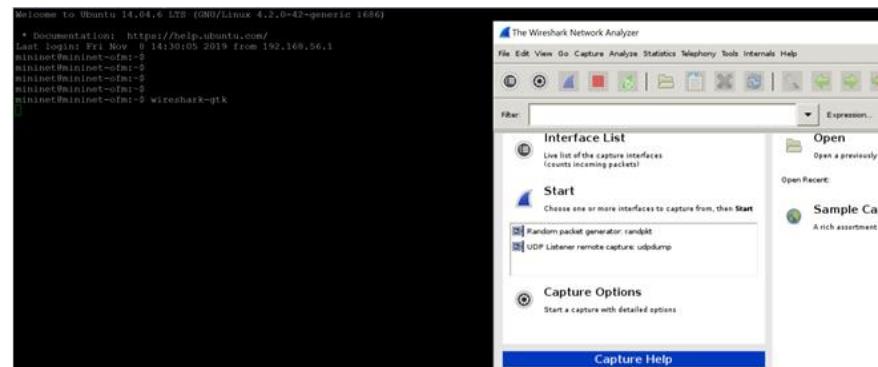


Validation and Verification for Packet Analyzer

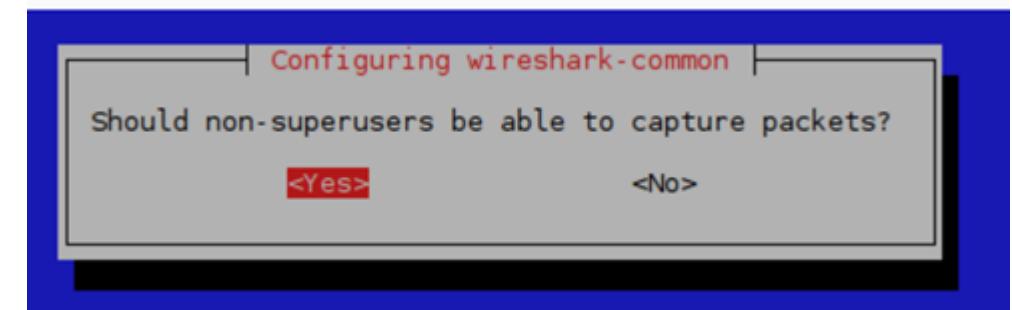
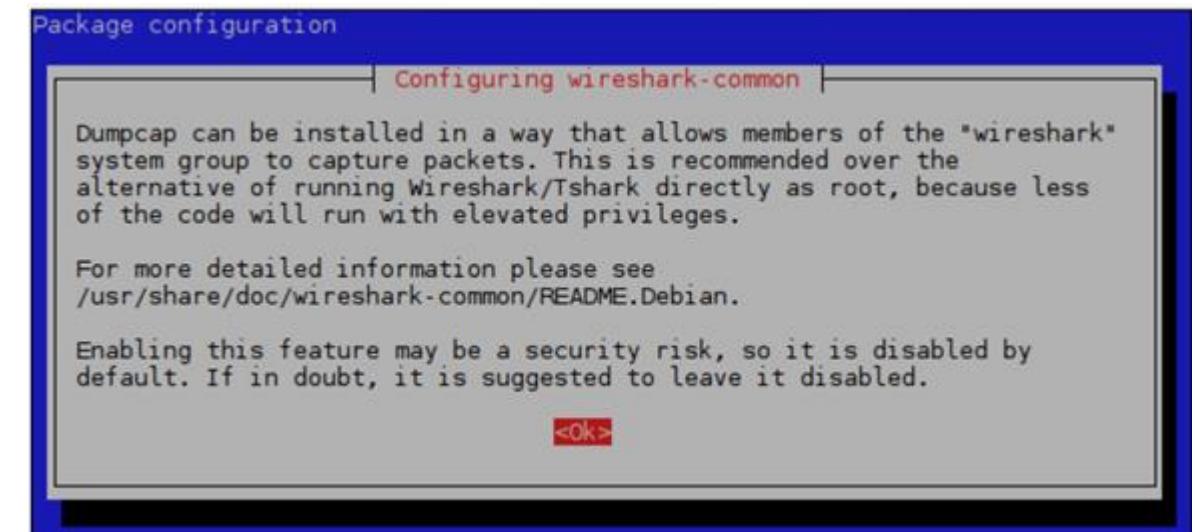
Supports various OpenFlow versions



Ease to run application from the terminal, great for automation

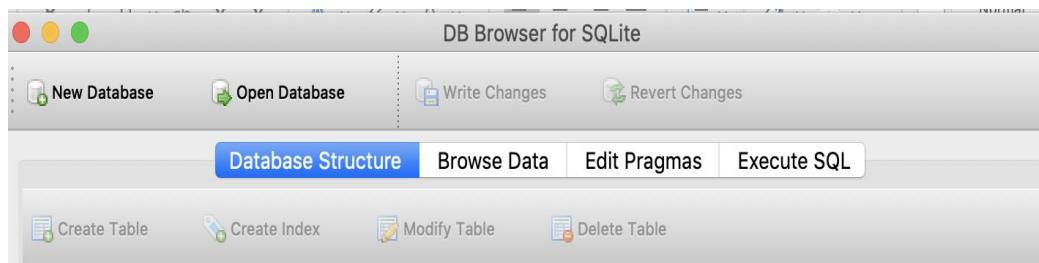


Superuser privilege to capture raw packets, hence more secure



Validation and Verification for Database

User friendly GUI for easy management



Easy application setup and schema definitions

Name	Type	Schema
Tables (5)		
appears		CREATE TABLE "appears" ("appearsid" int(11) NOT NULL , "bookid" int(11) NOT NULL , "characterid" int(11) NOT NULL)
books		CREATE TABLE "books" ("bookid" int(11) NOT NULL , "title" varchar(120) DEFAULT NULL , "storyid" int(11) NOT NULL)
characters		CREATE TABLE "characters" ("characterid" int(11) NOT NULL , "name" varchar(120) NOT NULL)
pictures		CREATE TABLE "pictures" ("pictureid" int(11) NOT NULL , "url" varchar(512) NOT NULL , "characterid" int(11) NOT NULL)
stories		CREATE TABLE "stories" ("storyid" int(11) NOT NULL , "story" varchar(120) DEFAULT NULL , "bookid" int(11) NOT NULL)
Indices (6)		
appears_bookid_idx		CREATE INDEX "appears_bookid_idx" ON "appears" ("bookid")
appears_characterid_idx		CREATE INDEX "appears_characterid_idx" ON "appears" ("characterid")
books_storyid_idx		CREATE INDEX "books_storyid_idx" ON "books" ("storyid")
books_title_UNIQUE		CREATE INDEX "books_title_UNIQUE" ON "books" ("title")
pictures_characterid_idx		CREATE INDEX "pictures_characterid_idx" ON "pictures" ("characterid")
stories_title_UNIQUE		CREATE INDEX "stories_title_UNIQUE" ON "stories" ("story")

Risks and Mitigation - PDR

Risk	Mitigation Action
1. Controller Failure	Introduce a secondary controller
2. Host Container Failure	Spin up a backup container on test VM
3. Intermediate device failure	Use a wireless router for redundant connection between devices
4. Traffic Congestion on Controller Link	Backup secondary link for test traffic
5. Hypervisor Failure	Spin up a backup test bed on SDN Controller hardware
6. Network Control Calls	Implement proactive flows.