University of Colorado
Interdisciplinary Telecommunications Program
CYBR 5700 Graduate Projects I

Containerized, Intelligent Network Functions on Hosts

Concept Definition Document (CDD)

Approvals

	Name	Affiliation	Approved	Date
Customer	Dr. Levi Perigo	CU Boulder ITP		
Course Coordinator	Dr. Kevin Gifford	CU Boulder ITP		

Project Customers

Name: Dr. Levi Perigo
Email: Levi.Perigo@colorado.edu

Team Members

Name: Afure Martha Oyibo Email: maoy4508@colorado.edu	Name: Kiran Yeshwanth Email: kiye0434@colorado.edu
Name: Manesh Yadav Email: maya8094@colorado.edu	Name: Prarthana Shedge Email: prsh1271@colorado.edu
Name: Soumya Velamala Email: sove4216@colorado.edu	

1.0 Project Description

Service networks are becoming more prominent and intricate with the advancement of technology. Managing a traditional network is a cumbersome task for network administrators due to the ever-changing network architecture. Smartphones and the Internet of Things (IoT) are pushing the barrier even further. Traditional network architectures that have minimal network visibility are challenging to scale, operate, manage, and cannot cope with the rising demand. These networks often use vendor-specific hardware which is costly, lack flexibility, and have a short product life-cycle due to the rapid progress in innovation.

Software-Defined Networking (SDN) is shifting the paradigm by providing automated services resulting in better network management, more straightforward implementation, and flexible solutions. SDN and Network Function Virtualization (NFV) provide the network services as abstracted, orchestrated flows controlled through centralized software using Virtual Network Functions (VNFs). NFV Edge [1] is the capability to bring network services such as routing, firewall, Network Address Translator (NAT), Quality of Service (QoS), and voice services to the edge of the network. The current implementation of NFV on the Wide Area Networks (WAN) edge, and mobile edge clouds as a standalone NFV solution integrated with traditional networks or as an SDN-NFV solution.

The Containerized, Intelligent Network Functions on Hosts project focuses on simplifying the SDN architecture by deploying VNFs as customized containerized applications on the hosts. The containers have access to the kernel and operate on a single instance, which reduces the kernel overhead and enhances performance as compared to Virtual Machines (VMs). This solution reduces the latency, throughput, provides enhanced QoS, faster service delivery, easier adoption of DevOps practices, and improved backward compatibility. It provides better scalability, reduced network complexity, and enhanced flexibility at a reduced cost by leveraging compute resources available on the hosts [2]. The application layer, with the help of the SDN controller, can push the flow entries on the containerized VNF at the host as per the requirement. For example, Intrusion Detection Servers (IDS) can be installed on the hosts to block perilous packets from entering the network. Using a simple Layer2 (L2) backbone network reduces network complexity, and increases scalability, as shown in the Containerized, Intelligent Network Functions on Hosts Architecture in Figure 1.

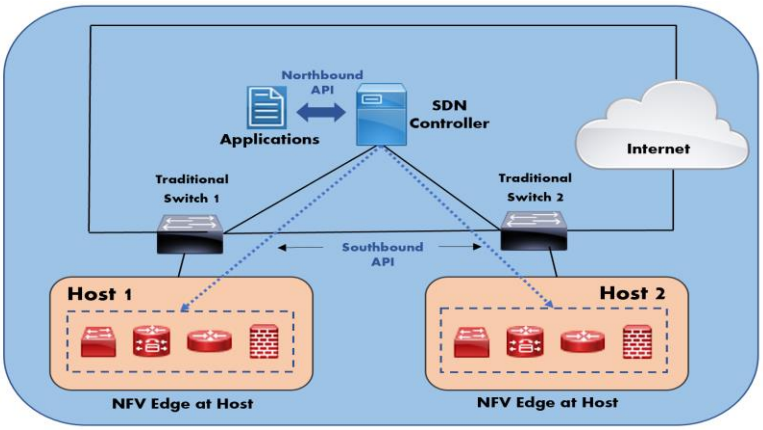


Figure 1: Containerized, Intelligent Network Functions on Hosts Architecture

The levels of success for the project are stated below:

3.1 Level 1 (Configuration Tests): Test the deployment of VNFs in the container(s). Create test cases to corroborate reachability to other devices in the network and to design test cases to validate flow entries and flow-table pipelining.

3.2 Level 2 (Performance Tests): Test the network performance parameters by subjecting it to various service tests. The network topology is to be subjected to different traffic using distinct service types, transmission rates and packet sizes to check QoS, and throughput. Evaluate the resource utilization of the containers to ensure that the host has sufficient capacity.

3.3 Level 3 (Failover Mechanism Tests): Test the network's ability to reallocate and use redundant resources after failures arising from device failures, link failures, network connectivity, congestion, improper configuration, and security breach. Verify the failover to conventional Transmission Control Protocol/Internet Protocol (TCP/IP) stack in case of SDN controller/VNF failure.

An Open vSwitch (OVS) VNF is created using containers on host devices, which supports flows for different network functionalities such as a switch, router, NAT, and a firewall. VNFs are to be created using containers to support standalone network functionalities such as Voice over IP (VoIP) and commercial firewall applications. The primary task of this project is to create different service chains using these VNFs and the flow table entries supported by them, as depicted in the Functional Block Diagram depicted in Figure 2. An SDN controller helps in automating, abstracting, orchestrating, and provisioning of the solution by using north and south-bound Application Program Interfaces (APIs). The application layer pushes the flow entries using the SDN controller to these VNFs as per the service chain scenario.

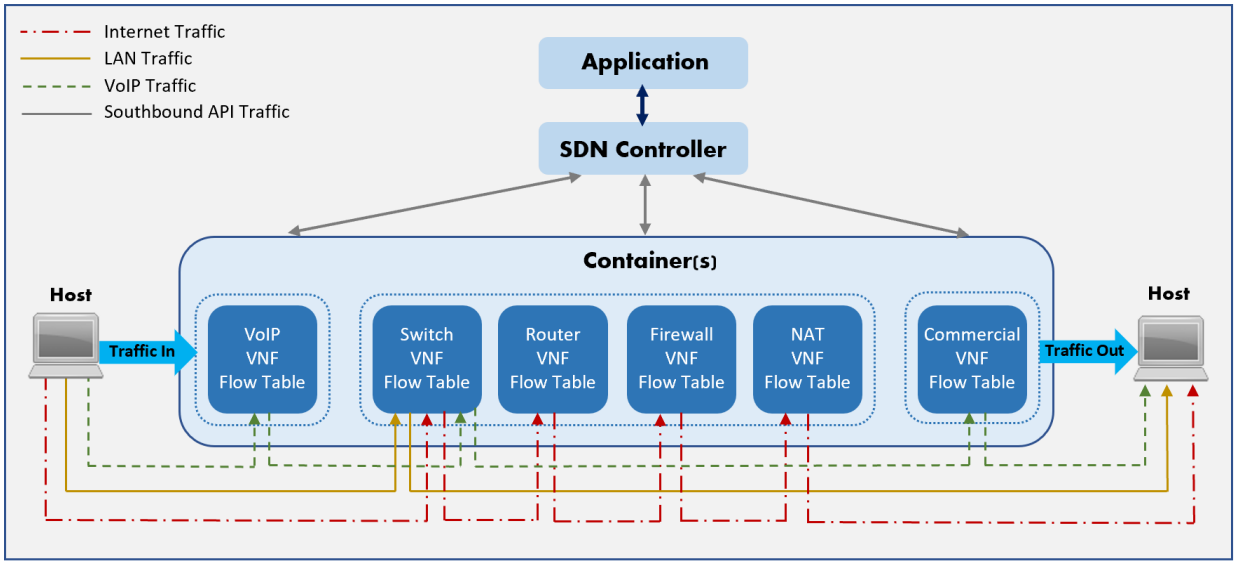


Figure 2: Functional Block Diagram (FBD)

A VM (Testbed VM) running on a hypervisor is used to create a testing framework to evaluate the deployed network for configuration, performance, and failover mechanism tests. A packet analyzer

is used at the switch interfaces to live capture the traffic to be used by the Testbed VM for analysis. The test results are sent to the application layer to store it in the database. With the help of a Data Visualization tool, the web interface displays the test results, as shown in the Concept of Operations (CONOPS) diagram in Figure 3.

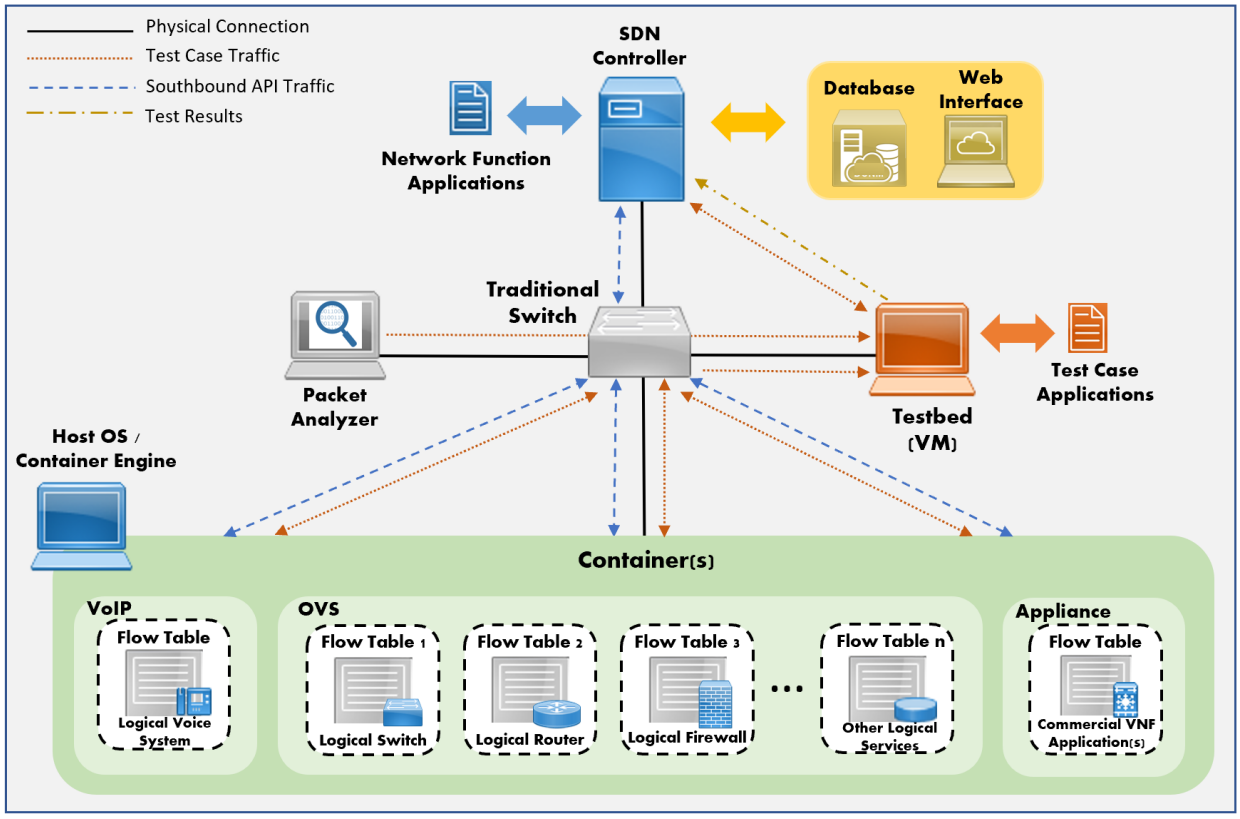


Figure 3: Concept of Operations (CONOPS)

Technical:

CPE 1.1 - VNFs creation: Create multiple VNFs corresponding to OVS, VoIP, and other commercial network services using containers running on the host**.**

CPE 1.2 - Configure and deploy SDN infrastructure: Deploy and configure SDN controller to install flows in the VNFs and manage the entire network. A packet analyzer is deployed on the edge switch to capture live traffic.

CPE 1.3 - Test VM creation: Emulate a test VM using the hypervisor to perform operational, performance, and functional tests.

CPE 1.4 - Service chain creation: Create multiple service chains with multiple VNFs to simulate and test different scenarios.

CPE 1.5 - Storage and display of test results: A database is created to store the test results in the application, perform data analysis by parsing it to a data visualization tool and display it on a web interface.

Logistical:

CPE 2.1 - Hardware devices: Using laptops/desktops with the SDN Controllers and SDN compatible switches in the Telecommunications laboratory for implementation.

CPE 2.2 - Containers/Controllers: Using open-source software and applications to create the network topology.
CPE 2.3 - Knowledge and concepts: Need to understand the functionality and uses of containers, SDN, VNFs to establish NFV Edge on a host. Determine a web Graphical User Interface (GUI) framework and learn the functionality to display the output from test results on the web interface.

2.0 Design Requirements

The functional requirements (FNC.X) and design requirements (DES.X.X) for Containerized, Intelligent Network Functions on Host project are listed below:

FNC.1: This project shall create multiple combinations of service chain using VNFs on the hosts
- **DES.1.1:** VNFs must be created for individual components in the service chain.
  - **DES.1.1.1:** OVS images must be used to create VNFs.
- **DES.1.2:** Containers shall be used to spin up VNFs.
  - **DES.1.2.1:** Virtual network interfaces must be used to connect the VNFs to form a service chain.
- **DES.1.3:** Containers shall be hosted on Windows and Linux Operating Systems (OS).

FNC.2: VNFs must be compatible with the outside network consisting of controllers and switches.
- **DES.2.1:** SDN controllers must have a customizable web-interface to install VNFs on the host devices by the push of a button.
  - **DES.2.1.1**: SDN controllers must support at least one northbound interface to allow the application layer to push intents to the controller
  - **DES.2.1.2**: SDN controller must support at least one southbound interface to push the flows as per the intents to the hosts.
  - **DES.2.1.3**: SDN controller must be able to send the notifications to the Test VM to perform the test cases on the hosts and obtain the test results.
  - **DES.2.1.4**: Deploy container manager application to initialize and manage the containers on the hosts.
- **DES.2.2:** Traditional switches must be used, which will perform the normal L2 forwarding at hardware speed.

FNC.3: Test cases to verify the configuration, performance and failover mechanisms for the service chain created.
- **DES.3.1:** A Test VM to host a test environment to run the test cases.
- **DES.3.2:** Well-defined, thorough test cases should be created to test the three scenarios mentioned above.
- **DES.3.3:** Different traffic types must be generated by the traffic generator tools to test the functionality of the VNFs.
  - **DES.3.3.1:** Test cases must be created according to various traffic parameters such as TCP, User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP), payload, encrypted and unencrypted traffic.

DES.3.3.2: Test cases must be able to check the VNFs for the performance parameters such as latency, throughput, jitters, delay, losses, memory utilization, and redundancy.

FNC.4: The output of the test cases must be stored and displayed on the web interface.

DES.4.1: A database manager to store and process the test results.

DES.4.2: A web-interface to display the test results in real-time.

3.0 Key Design Options Considered

Different design options were considered for the containers, SDN controllers, hypervisors, web-interface, database manager, and packet analyzer to satisfy the demand for Containerized, Intelligent Network Functions on Hosts project. The below analysis focuses on both open-source as well as commercial solutions available for these components.

SDN Controllers

SDN Controller [3] is the logically centralized, physically distributed application in the SDN architecture which manages flows in the network. Having a global view of the network, SDN controllers can provide more educated traffic manipulation, resulting in enhanced network performance and better management. As SDN is gaining popularity, many SDN controllers are now available in the market with improved features. Both open-source and commercial SDN solutions are available in the market. The available controller option is examined below in detail.

1. Open Network Operating System (ONOS):

ONOS [4] is an open-source SDN controller developed by Linux Foundation. It is a Java-based controller designed to provide a carrier-grade network to the service providers. A few salient features of ONOS are:

- Distributed Flat architecture
- Resiliency
- High availability
- Supports OpenFlow version 1.0 to 1.3 as a southbound protocol
- High performance on large scale networks

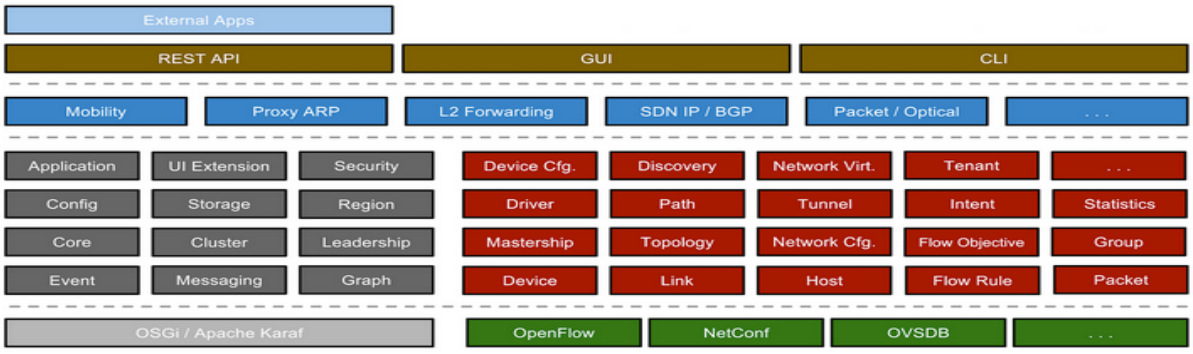


Figure 4: ONOS Architectural Framework [5]

Pros and Cons:

Pros	Cons
Well documented	Not suitable for cloud computing and data center architecture
High scalability	Not compatible with legacy network
Interactive GUI framework which can be easily customized as per the requirement	Provides sub-optimal performance when installed on Windows OS [42]
Enhanced security features	
Supports multiple southbound protocols such as OpenFlow, P4, Network Configuration Protocol (NETCONF)	
Regular updates available	
Provides northbound abstraction using REST, gRPC or native interface	

Table 1: Pros and Cons of ONOS SDN Controller

2. OpenDaylight (ODL):

ODL [6] is an open-source, Java-based SDN controller hosted by Linux Foundation. It is designed to promote SDN and NFV technologies. A few salient features of ODL are:

- Distributed Flat architecture
- Supports many southbound protocols such as OpenFlow and NETCONF
- Supports OpenFlow version 1.0 to 1.3 as a southbound protocol
- Provides maximum flexibility to users in terms of requirements
- Works on all Linux Operating Systems

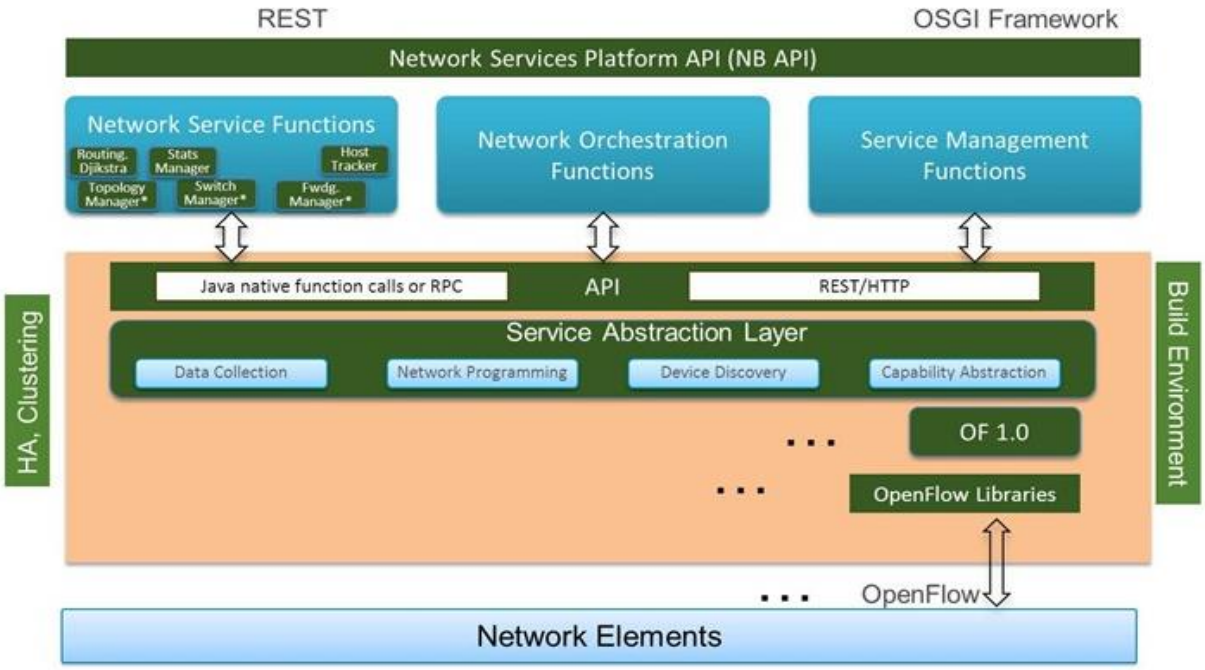


Figure 5: ODL Architectural Framework [7]

Pros and Cons:

Pros	Cons
Enhanced security features	Does not scale well for large networks (Scalability)
Supports multiple southbound protocols such as OpenFlow, P4, NETCONF	Complex features and functionalities which requires more time and efforts to understand
Provides multiple ways to integrate with OpenStack	Proper documentation not available
Compatible with range of legacy devices/networks	Not suitable for carrier-grade networks
High performance on cloud computing and data center networks	Performance varies from feature to feature

Table 2: Pros and Cons of ODL SDN Controller

3. Ryu:

Ryu [8] is an open-source SDN controller designed by NTT. It is a Python-based controller aimed at providing a component-based SDN framework. A few salient features of Ryu are:

- Centralized architecture
- Supports multiple southbound protocols such as OpenFlow, NETCONF
- Supports OpenFlow version 1.0 to 1.5 as a southbound protocol
- Code is available under Apache 2.0 license

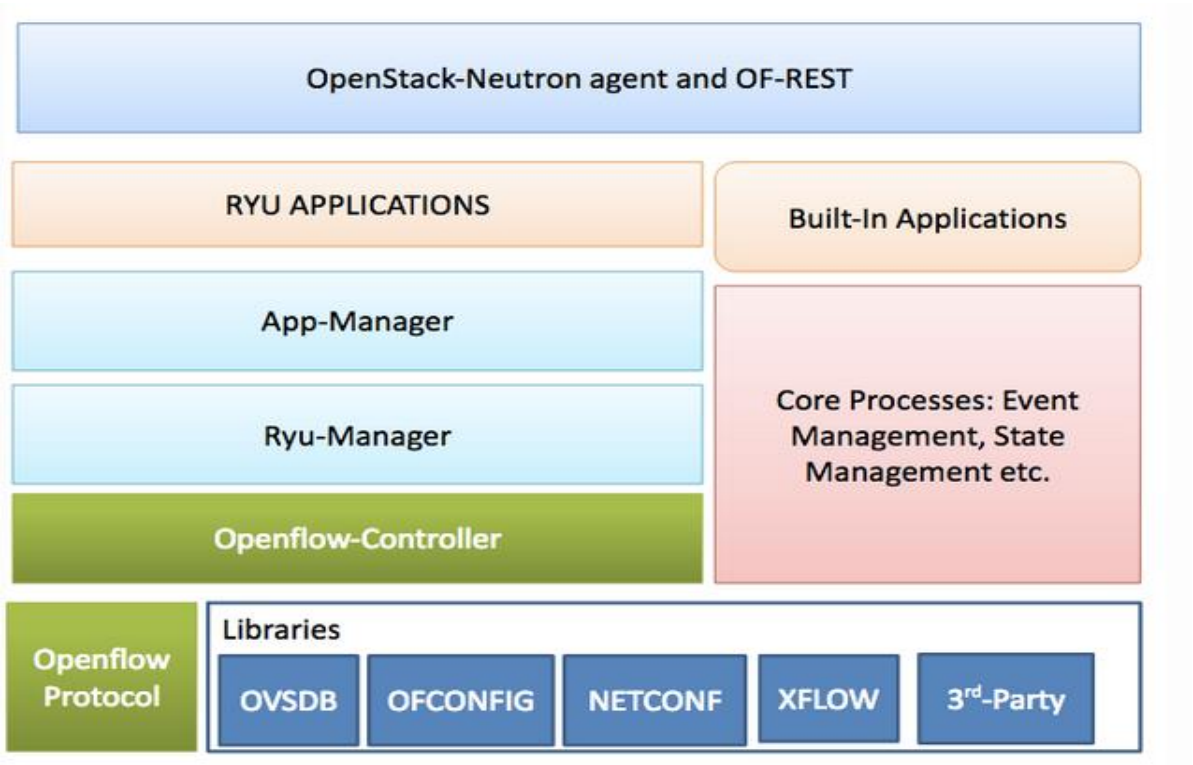


Figure 6: Ryu Architectural Framework [9]

Pros and Cons:

Pros	Cons
Provides component-based applications which can be customized according to the requirement	GUI is not intuitive Can be implemented on Linux OS, MacOS but not on Windows OS [42]
Simplified features and functionalities which can be understood easily with less time and efforts	Poor language runtime efficiency
Integrated with applications such as OpenStack Quantum, and IDS	Lacks support for features such as Infrastructure as a Service (IaaS), Security
Supports Rest API as well as other user-defined northbound APIs	Scalability issues
Well documented	Complex testing method

Table 3: Pros and Cons for Ryu SDN Controller

4. Floodlight:

Floodlight [10] is an open-source, Java-based SDN controller developed by an open community of developers. It is forked from the Beacon controller which was developed at Stanford University. Big Switch Networks offered floodlight to the open-source community as a part of ODL project. Below are a few features of the Floodlight controller:

- Centralized architecture
- Supports OpenFlow version 1.0 to 1.5 as a southbound protocol
- Compatible with physical and virtual switches
- Supports OpenStack cloud Orchestration platform

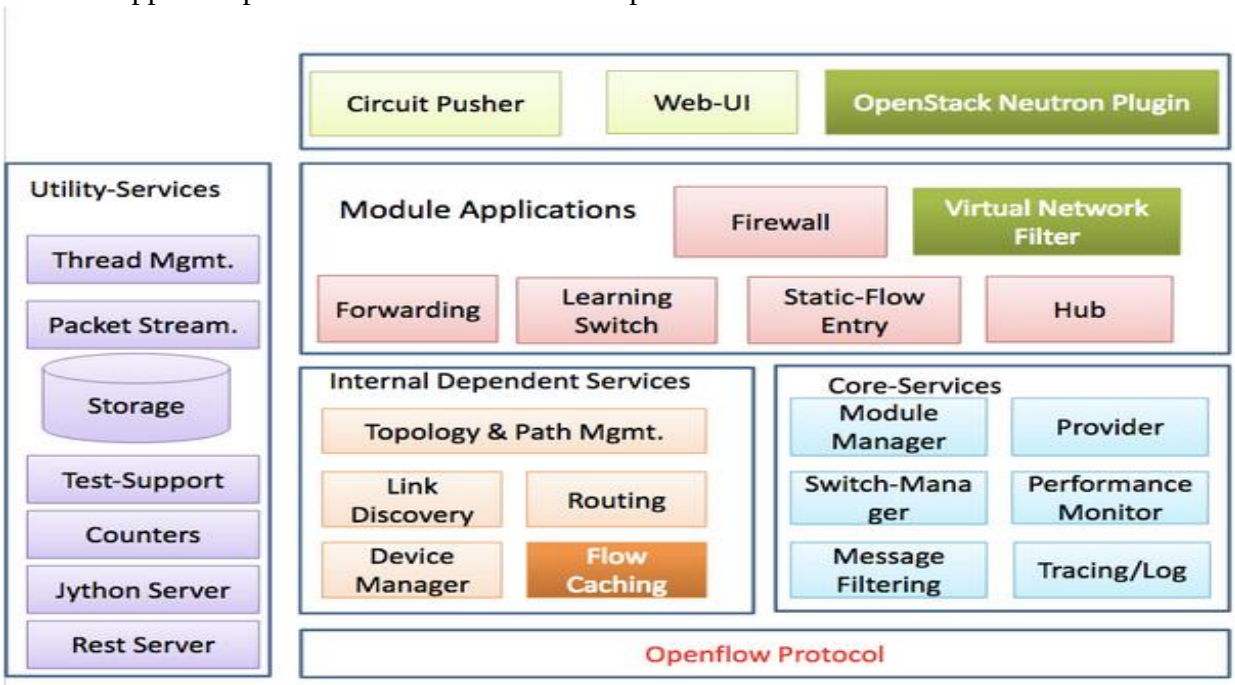


Figure 7: Floodlight Architectural Framework [11]

Pros and Cons:

Pros	Cons
Faster deployment	Rest API is slow which makes real time reaction to changes difficult
Well documented	Complex features and functionalities which requires more time and efforts to understand
Interactive GUI framework which can be easily customized as per the requirement	Limited Configuration management
Provides good debugging	Configuration is lost at reboot
Integrated with applications such as Firewall, Virtual Network Filter	Lacks high availability
Efficiently manage OpenFlow and non-OpenFlow environments simultaneously	

Table 4: Pros and Cons of Floodlight SDN Controller

5. Cisco Application Policy Infrastructure Controller (Cisco APIC):

Cisco APIC [12] is a commercial SDN controller developed by Cisco which acts as a single point of policy and management for Cisco devices. A few features of the Cisco APIC are:

- Centralized architecture
- Integrates well with Cisco products
- High availability
- Agile
- A good solution for cloud computing and data center networks
- Enhanced service delivery

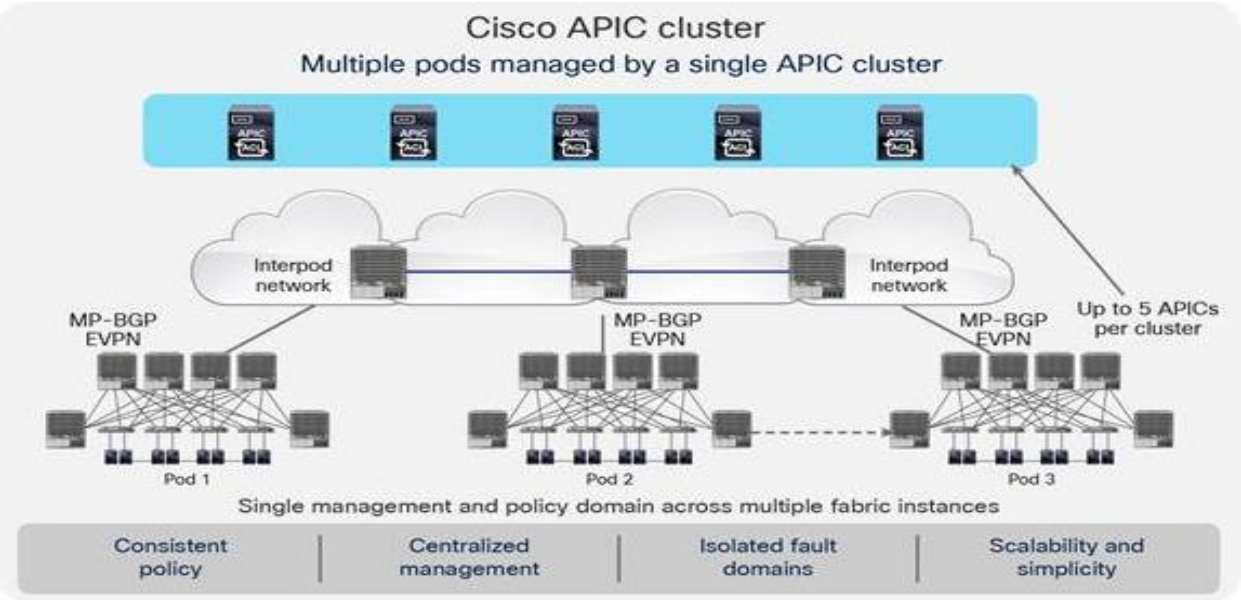


Figure 8: Cisco APIC Architectural Framework [13]

Pros and Cons:

Pros	Cons
Well documented	Vendor lock-in
Enhanced security	Lacks interoperability with other vendor devices
Scalability	Costly
Faster service delivery	Works with Cisco Nexus 9000 hardware
Real time visibility and application heath support	
Unified storage, computation and networking environment	
Improved self-provisioning and Orchestration	

Table 5: Pros and Cons of Cisco APIC SDN Controller

Containers

Containers [14] are defined as the standalone, executable software packages for an application comprising code, user information, and other dependencies. Containers being lightweight takes fewer system resources and directly operates on the host OS. Containers help the applications to run uniformly on different environments by isolating the application from the host environment. VM [15], consume many system resources and runs on a separate OS. Due to this, containers provide faster deployment, simplified security updates along with reduced management overhead which gives them an advantage over VMs. Containers are gaining popularity in the SDN domain as they help in streamlining network functionalities and operations. The available container options shall be discussed in more detail:

1. Docker:

Docker [16] is an open-source container tool that uses OS-level virtualization to provide Platform-as-a-Service (PaaS) to applications in different environments. Docker engine is used to host and manage the containers. Docker stores the application codes and other related libraries in a separate container image file. This separation streamlines the rollout process from development and testing to deployment and integration.

Docker was developed by Docker, Inc. in 2013, as an enhancement to Linux Containers (LXC). Their goal was to build single-application containers that can provide rapid deployment and enhanced scalability. Below are a few features of Docker [17]:

- Open-source with large community support
- Efficient use of same container image from development to deployment
- Nearly bare-metal execution speed
- Execution space isolation
- Security Management
- Increased productivity
- Provides a platform for fast pace innovation

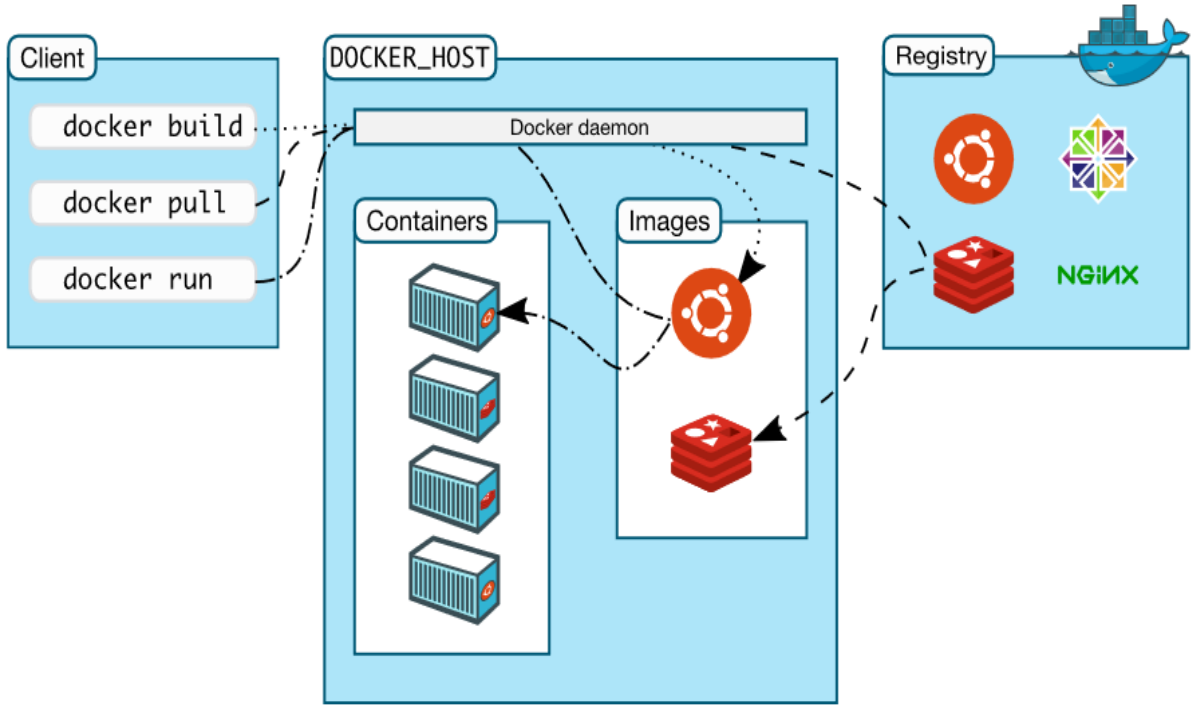


Figure 9: Docker Architectural Framework [18]

Pros and Cons:

Pros	Cons
Well documented	Poor monitoring
Lightweight	Does not perform well with GUI applications
Suitable for continuous deployment and integration (Scalability)	Complex features and functionalities which requires more time and efforts to understand
Easy debugging	Lack of storage options
Can be deployed on Linux and Windows OS	Less secure- assigns root privileges to all applications
Frequent updates	

Table 6: Pros and Cons of Docker Container

2. CoreOS Rocket:

Rocket [19] is an open-source container tool developed by CoreOS in 2014. It was aimed at providing better security and operation for container deployment. Rocket containers use image signature validation and distributed deployment approach to secure the containers from the other containers and the host OS. Below are a few features of Rocket:

- Open-source
- Image signature validation
- Reduces the risk of version compatibility problems
- Good community support

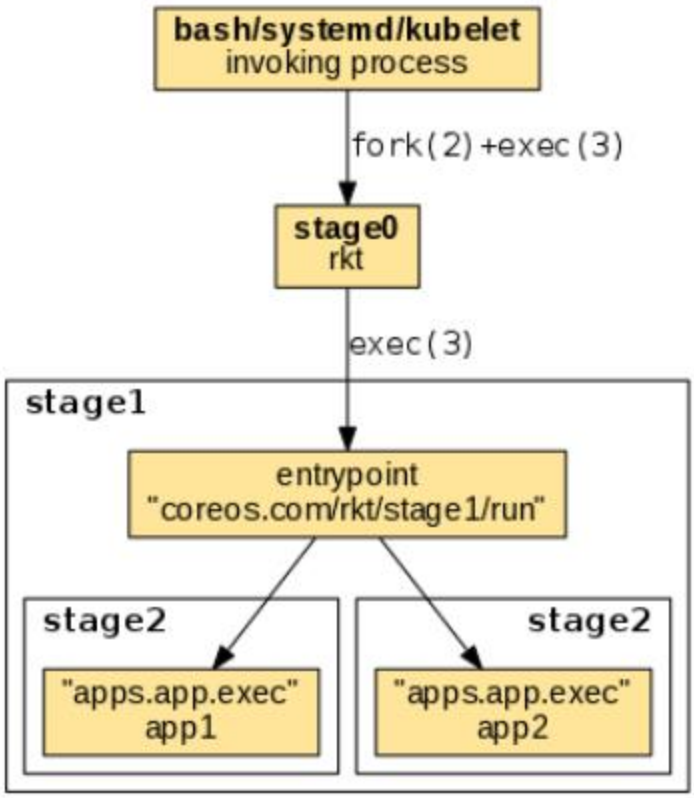


Figure 10: Rocket Architectural Framework [20]

Pros and Cons:

Pros	Cons
Well documented	Limited scope for non-Linux based environment
Lightweight	Limited support for third party applications
Runs the new containers as unprivileged users making the environment secure	
Intuitive Command Line Interface (CLI)	
High scalability	

Table 7: Pros and Cons of Rocket Container

3. Solaris Containers:

Solaris Containers [21] is an open-source container tool designed by Sun Microsystems in 2005 for OS-level virtualization. It uses zones for boundary separation and resource control. Zones are the virtual server instances with dedicated operating systems. It is compatible with x86 and Scalable Processor Architecture (SPARC) systems platforms. Below are a few features of Solaris containers:

- Open-source
- Easy up-gradation [22]

- Reduces risks
- Enhanced debugging
- Enhanced security by using zones

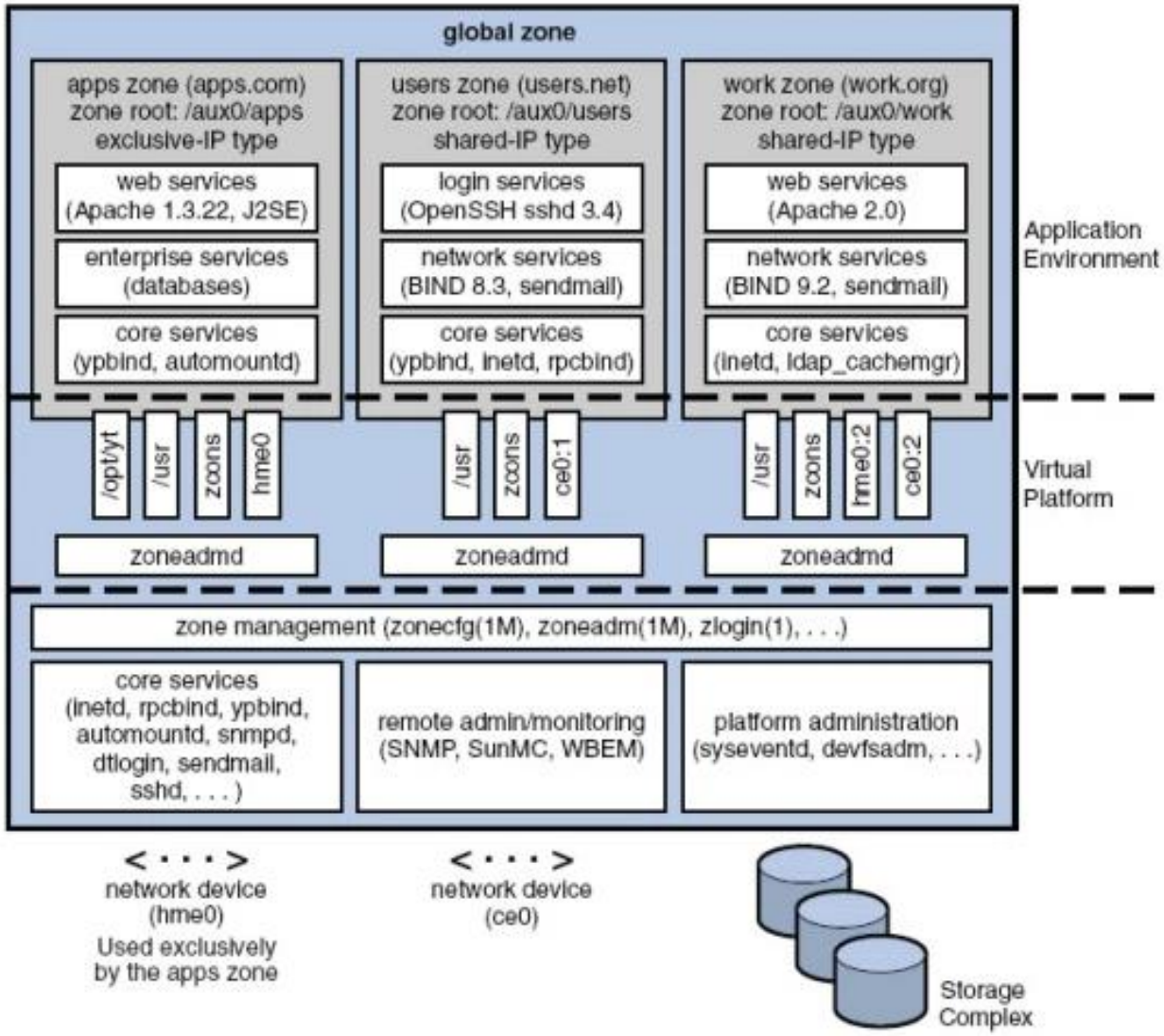


Figure 11: Solaris Containers Architectural Framework [23]

Pros and Cons:

Pros	Cons
Well documented	Scalability
Lightweight	Takes time to deploy
Better data storage	Does not work on non-Linux environments
Easy management	Compatible with Oracle containers only
Faster deployment	

Table 8: Pros and Cons of Solaris Container

Packet Analyzer

Packet analyzer [24] is software that is used to capture and perform packet analysis of the network traffic. This software or tool can intercept the traffic passing through the network with which network issues, security concerns, network health can be managed and maintained. With the help of a packet analyzer, complete visualization of the status of bandwidth and resource utilization can be done. The following are a few critical functions of a packet analyzer:

- Analyze issues concerning the network
- Network monitoring for unauthorized network access
- Monitor network bandwidth and per-user bandwidth consumption
- Availability of network statistics
- Filtering unwanted traffic
- Prevention of unauthorized access
- Troubleshoot network operational issues
- Perform end-to-end network management with the data obtained

1. Wireshark:

Wireshark [25] is the most widely used, open-source packet analyzer that supports a multitude of protocols and compatible with almost every popular OS. It is an excellent option for network troubleshooting and performance analysis. It is a lightweight software that runs with minimal processing power.

Pros and Cons:

Pros	Cons
Lightweight and user-friendly GUI	Capture files becomes extremely large, making arduous to identify required packets
Available for all the flavors of OS	
Supports many protocols, allows the capture on multiple interfaces at once	

Table 9: Pros and Cons of Wireshark Packet Analyzer

2. Tshark:

Tshark [26] is the CLI version of Wireshark, which captures and displays packets on the terminal. It provides the same support as Wireshark. Despite not have an interactive user interface, the decoded output displayed on the terminal is more human-readable. Hence Tshark can be easily leveraged for automation.

Pros and Cons:

Pros	Cons
Terminal based interactive tool	No GUI
Can be leveraged in network automation	
Supports OpenFlow for SDN packet analysis	

Table 10: Pros and Cons of Tshark Packet Analyzer

3. Tcpdump:

Tcpdump [27] is a CLI utility used to isolate traffic with a variety of options like ports, protocols, IP, and other layers. It is a general-purpose packet sniffer that is available on most of the Linux distros by default and performs actions based on options and expressions. Tcpdump is very useful in real-time, quick inspection scenarios; however, not recommended for in-depth packet analysis.

Pros and Cons:

Pros	Cons
Terminal based interactive tool	No GUI
Can be leveraged in network automation	Limited protocol support
Quick and easy to use	

Table 11: Pros and Cons of Tcpdump Packet Analyzer

Hypervisors

A hypervisor is a software, firmware, or hardware that enables multiple OSs to share the same hardware. It controls the allocation of a portion of the physical host hardware and kernel resources such as processor cycles, memory space, network bandwidth, disk space to each virtualized OS.

There are two types of hypervisors [28]:

Type 1: This type of hypervisor runs directly on the system hardware. They are embedded in the physical host bare metal and therefore provide better throughput.
Type 2: This type of hypervisors runs on the physical host OS which provides virtualization services such as Input/output device support and memory management. Type 2 hypervisor provides better control for OS-dependent virtualized applications.

Figure 12 shows the architecture of the two types of hypervisors [29].

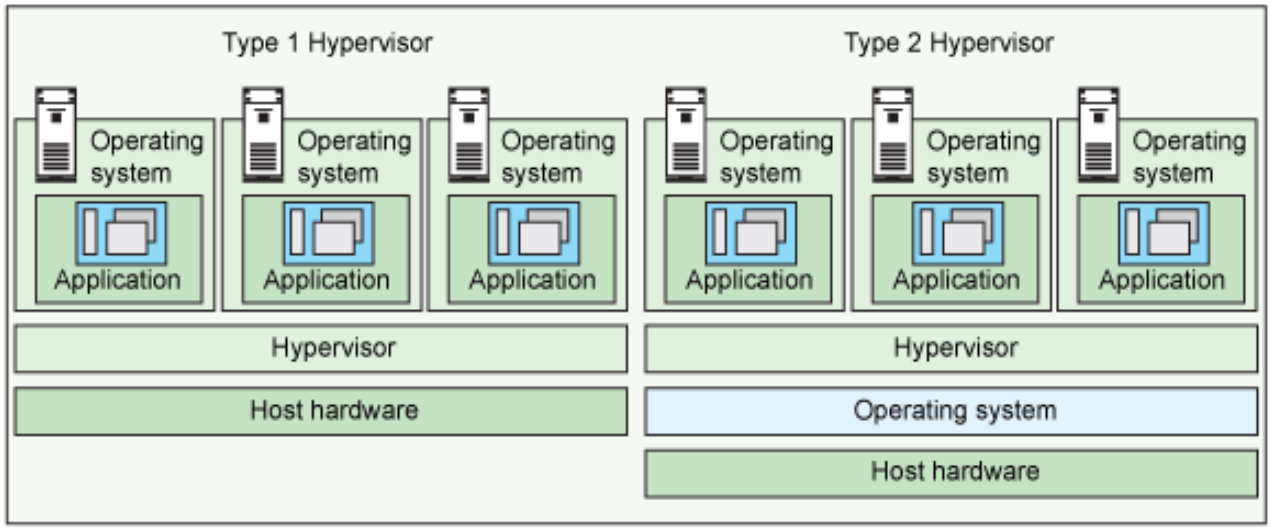


Figure 12: Hypervisor Architectural Framework [30]

Below is the analysis of Type 1 hypervisors:

1. VMware ESXi:

VMware ESXi is an OS-independent hypervisor that runs directly on bare-metal hardware. It comprised of a console and system dashboard for management. It supports large scale production features like live migration, load balancing, and pooling of storage and computes resources among multiple hosts.

Minimum system requirements [31]:

- Supports only 64-bit x86 Central Processing Unit (CPU) with at least two cores
- Minimum of 2GB of Random Access Memory (RAM), 8GB of RAM recommended
- Hardware virtualization- Intel Virtualization Technology (VT-x) or Advanced Micro Devices (AMD) Rapid Virtualization Indexing (RVI) support
- One or more Gigabit or 10Gb Ethernet controllers

Pros and Cons:

Pros	Cons
Advanced set of large-scale virtualization deployment features	Licensed product
GUI and CLI for management	Slow VM initialization setup and performance
	Steep learning curve

Table 12: Pros and Cons of VMware ESXi Type 1 Hypervisor

2. Kernel-based Virtual Machine (KVM):

KVM is Linux-kernel based hypervisor. It is the best solution for virtualizing Linux environment. It also provides an efficient layer of virtualization because it retains the performance of the bare metal it is running on.

Minimum system requirements [32]:

- One core or thread for each virtualized CPU
- Minimum of 2GB RAM
- Hardware virtualization (Intel VT-x or AMD RVI) support
- One or more Gigabit or 10Gb Ethernet controllers
- 6GB free hard disk space

Pros and Cons:

Pros	Cons
Better support for network virtualization	Poor support for large scale storage virtualization
Free and open-source	
Cheaper to implement	
Enhanced VM security and isolation	

Table 13: Pros and Cons of KVM Type 1 Hypervisor

3. Microsoft Hyper-V (MS Hyper V):

MS Hyper V is a server virtualization product from Microsoft. It can operate as both a type 1 and type 2 hypervisor. It offers a unified set of integrated management tools that allow easy administration and deployment of virtualized workloads.

Minimum system requirements [33]:

- 64-bit processor with Second-Level Address Translation (SLAT)
- 4 GB of RAM
- 32 GB hard drive space
- 1GB Ethernet controllers
- Hardware virtualization (Intel VT-x or AMD RVI) support

Pros and Cons:

Pros	Cons
Better support for network virtualization	Licensed product supports a limited number of guest OS choices
Hypervisor maintenance does not result in downtime	
Faster VM initialization setup	
Support type 1 and type 2 hypervisor deployment	

Table 14: Pros and Cons of MS Hyper V Type 1 Hypervisor

Below is the analysis of Type 2 hypervisors:

1. VMware Workstation:

VMware Workstation is a robust hypervisor with some advanced features for running multiple OSs or versions of one OS on one desktop, virtualizing sandbox environments and snapshots, labs, and demonstration purposes. It provides a secure and isolated environment for virtualized resources. It is commercial software that requires the purchase of a license for usage and support.

Minimum system requirements [34]:

- 64-bit x86 Intel or AMD Processor from 2011 or later
- 1.3GHz or faster core speed
- 2GB RAM minimum required
- 4GB RAM recommended
- 1.2 GB of available disk space for the application
- Additional hard disk space required for each virtual machine

Pros and Cons:

Pros	Cons
Advanced set of virtualization features	Licensed product
User friendly GUI and CLI for management	Different versions of product for different host operating systems
Better security features for virtualize resource	

Table 15: Pros and Cons of VMware Workstation Type 2 Hypervisor

2. VirtualBox:

VirtualBox is a portable, free, and open-source hypervisor that provides consistent performance for a small-scale virtualization need. It is easy to install and utilizes a small amount of host system resources for its operations.

Minimum system requirements [35]:

- 64-bit and 32-bit Intel or AMD Processor from 2011 or later
- 512GB RAM minimum recommended
- Integrated Development Environment (IDE), Serial Advanced Technology Attachment (SATA), and Small Computer System Interface (SCSI) hard drives supported
- Additional hard disk space required for each virtual machine

Pros and Cons:

Pros	Cons
Free, open-source software managed by Oracle	Limited security features for virtualized resource
Cross platform solution for any host OS	
User friendly GUI and CLI for management	

Table 16: Pros and Cons of VirtualBox Type 2 Hypervisor

3. QEMU:

QEMU is a host OS hypervisor that performs hardware virtualization. The code execution can occur on various processors like ARM on x86 or PPC on ARM. It also emulates Virtual Machine (VM) resources, but the performance is extremely slow as all the emulation performed is in software.

Minimum system requirements [36]:

- Any x86 64-bit and 32-bit hardware
- KVM as an accelerator for physical CPU virtualization extensions
- Linux host
- Minimum of 2GB RAM

Pros and Cons:

Pros	Cons
Suitable for applications that requires the emulation of specific hardware types	Slow emulation of VMs
Open-source product	No GUI for management
	Low performance and reliability

Table 17: Pros and Cons of QEMU Type 2 Hypervisor

Database Storage

Databases are logical structures used to store and index data. It is a collection of data that can be accessed, modified, and updated by computer software. In this project, the database is used to collect network data that would be accessed by our GUI to present a network monitoring/ visualization dashboard.

There are two types of database:

1. **Non-relational database:** are databases that do not store data based on a predefined structure. Depending on the type of data stored, the database falls into several categories. These include a key-value store for storing key-value pairs, a wild-column store for storing multidimensional key-value pairs; a document store for storing data in JSON format; a graph database for storing related nodes of an object. Non-relational databases are easier to administer because they have no schema. However, they do not provide data consistency and are not as widely adopted as relational databases.

2. **Relational databases:** These are databases that store data based on a predefined category called a schema. It uses a set of tables where the row defines an instance, and the column defines an entry for data in a specify categories associated with the instance as defined by the schema. The Structured Query Language (SQL) is the standard user and application program interface for a relational database. In ordered to ensure contained data, relational databases provide data integrity by using several constraints in the tables. For this reason, we shall be adopting a relational database.

While there are many relational databases, these are the most widely adopted relational databases:

1. Oracle:

Oracle is a proprietary commercial relational database. All platforms support it. It has Oracle's Help Center, which provides useful information, including getting started guides and advanced features. It provided support for storing multi-model data like document, graph, relational, and key-value. It is best suited for large scale integration as it is license cost.

Minimum Requirements:[37]

- Minimum 500 MB free disk space for installation, 10 GB recommended
- Minimum of 1GB physical memory, 4GB recommended
- AMD Opteron, Intel Pentium at 500 MHz or faster, or Intel EM64T

Pros and Cons:

Pros	Cons
Better security features	Licensed product
Provides a high degree of scalability	Suitable for large scale deployment
	Closed sourced product not easily customizable

Table 18: Pros and Cons of Oracle Database

2. MySQL:

MySQL is an open-source relational database based on SQL. It uses tables, constraints, triggers, roles, stored procedures, and views as the core components. It is supported on all platforms and has an open and commercial community support options as well as official documentation. It is suitable for large and small-scale integration.

Minimum Requirement [38]:

- 2 CPU Cores
- 2 GB RAM
- Disk I/O subsystem applicable to a write-intensive

Pros and Cons:

Pros	Cons
Free product	Non-lightweight application
Suitable for any size of deployment	
Easily customizable to meet user requirements	
Support almost all datatypes	

Table 19: Pros and Cons of MySQL Database

3. SQLite:

SQLite is an open-source SQL database that stores data to a text file. It supports all the relational database features and is suitable for portable integrations where applications need to write data directly to a disk. However, it is not suitable for high-performance driven operations.

Minimum Requirements [39]:

- Dual-Core 2GHz or higher
- 4 GB RAM
- 1 GB free disk space

Pros and Cons:

Pros	Cons
Lightweight application	Support only one connection at a time
Limited data type support	Performance degrades with large set of data
Support for multiple connections	No security feature

Table 20: Pros and Cons of SQLite Database

Traffic Generators

Traffic generators are tools that enable the generation of customized traffic and monitoring of the performance parameters of the network. They are used to check the robustness of the network under various scenarios before deployment and to recreate failure cases. These tools emulate a client-server model to initiate traffic, monitor it, and register the metrics on each segment of the path for analysis.

Traffic generators can be categorized based on:

- **Network traffic model:** These type of generators generates traffic based on typical network models based on their traffic distribution calculations.
- **Traffic characteristics:** These constitute of flow-based and packet-based traffic generators. Flows are similar applications or protocols that traverse end-to-end. Flow-based traffic generators generate traffic based on flow characteristics of the first network. A packet is the network layer entity of a network. Packet-based traffic generators replicate its characteristics such as packet size and arrival process in the first network based on recordings of traffic streams of the existing network.
- **Application protocols:** These generators can be defined to generate flows for web, email, chat, and other such high-level applications. They can also generate coarsely defined traffic for specific applications like Skype and P2P-TV [40].

1. Ostinato:

Ostinato is a multi-stream open-source packet generator that uses powerful Python APIs to generate streams of traffic. It serves as a diagnostic tool for networks employing legacy products in conjunction with high-end servers. It has an interactive GUI that can be launched as a Python app through an API or via an executable [41]. It supports all common standard protocols such as Ethernet, ARP, IPv4, IPV6 and much more.

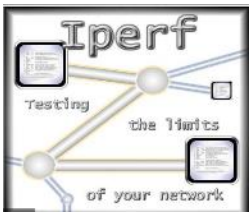
Pros and Cons:

Pros	Cons
Easy to install and implement	The traffic analytics are basic
User-friendly GUI	Does not support application-based traffic generation
Supports both IPv4 and IPv6 traffic	Poor documentation
Free of cost	Slower bug fixes
Support for multiple streams to be sent simultaneously	

Table 21: Pros and Cons of Ostinato Traffic Generator

2. iPerf:

iPerf is a traffic generation tool that can be used to measure the maximum network throughput and quality of a network connection between two separate servers. Iperf is available for all Linux distribution and is available for Windows as well. It employs a client-server model to send and receive acknowledgments of traffic along with the complete characteristics of the intermediate network [42].



Pros and Cons:

Pros	Cons
Easy to install and implement across multi-vendor platforms	Cannot monitor CPU usage and memory utilization of intermediate nodes
Free of cost	GUI support is not great
Supports both IPv4 and IPv6 traffic	With multiple streams, dedicated scripts are necessary to connect client to server
Huge number of deployments in the industry leading to better support	Does not support L7 signature-based traffic generation
Good documentation	
The results are saved in easily readable log files	
Supports multithreading	

Table 22: Pros and Cons of IPerf Traffic Generator

3. Ixia

Ixia traffic generators are hardware tools that support the testing of a wide range of performance-heavy networks and conform to the RFC 2544 traffic generators benchmark. The tools are commonly deployed in data centers at the end-points or customer-facing equipment to generate traffic with varied latencies, packet sizes and Layer 7 (L7) signatures and perform detailed traffic analytic reports on received performance indicators. IxExplorer, the GUI provided by Ixia manages the testing hardware and provides control over-analysis of over 100Gigabytes of traffic at any time [43].

Pros and Cons:

Pros	Cons
In addition to end-to-end analysis, it provides the status of the health of backbone network	Steep learning curve, no interactive shell
A wide range of supported L2 to L7 protocols	Cost of deploying the hardware is high
Traffic generation can be automated and made to respond to pre-defined events or anomalies	Poor documentation
Great GUI support	
The hardware is scalable and compatible with high density server applications	

Table 23: Pros and Cons of Ixia Traffic Generator

4. WAN Killer

It is a software traffic generator by SolarWinds that allows the generation of both highly specific custom traffic and randomized traffic based on user requirements. It checks TCP and UDP packets based on randomized traffic before the test cases are run to check the operating conditions under typical environment. The tool comes with a widely compatible light-weight GUI that helps drill down to specific keys like Differentiated Serves Code Point (DSCP), as well as Explicit Congest Notification (ECN) settings [41][43].

Pros and Cons:

Pros	Cons
Supports generation of very specific traffic based on DSCP that is supported by very few other generators	Commercial tool comes at higher price
The GUI is intuitive and easy to use	Limited support for application protocols like File Transfer Protocol (FTP) and Simple Mail transfer Protocol (SMTP)
Great documentation	
Faster bug fixes	
The GUI is intuitive and easy to use	

Table 24: Pros and Cons of WAN Killer Traffic Generator

4.0 Trade Study Process and Results

For the success of the project, it is crucial to analyze and weight critical project elements across various factors. This weighted comparison helps identify the most desirable and appropriate elements for the project. Parameter scoring is done on a scale of 1 to 5. Table 25 lists various parameters used to analyze the SDN controller, traffic generator, database storage, and packet analyzer technologies. Similarly, Table 26 lists various parameters used to analyze container technology and Table 27 lists various parameters used to analyze hypervisor technology.

Parameter	Description	Weight	Scale
Cost	Cost is the most important factor. Team aims at incurring minimum cost of deployment and operation for this project.	0.20	5 – Low Cost 1 – High Cost
Open-source	Open-source nature of the technology allows programmers and developers to leverage other's work and customize it one's need.	0.15	5 – Openness 1 – Proprietary
Feature support	The parameter covers the technologies ability to support various Northbound and Southbound APIs to installing new features and applications at the push of a button.	0.10	5 – Multiple Features 1 – Few Features

Programmability	The technology should be easy to program and should support the most popular automation programming language. The technology should also provide good API support.	0.15	5 – Lucid 1 – Complex
Community Support & Documentation	Good community support and documentation help in easy bug fixes and in deploying technology features. A good online community provides almost 24x7 support for most issues.	0.15	5 – Superior Support 1 – No Support
Learning curve	Technology should be easy to learn and implement	0.10	5 – Lucid 1 – Complex
Modularity	The ability to customize the given technology to suit project needs is critical. We shall be testing various scenarios to see how our network functions behave.	0.15	5 – Easy to Modulate 1 – Challenging to Modulate

Table 25: Trade study Parameters, Weight and Reasoning for SDN controller, traffic generator, database storage, and packet analyzer technologies

Parameter	Description	Weight	Scale
Cost	Cost is the most important factor. Team aims at incurring minimum cost of deployment and operation for this project.	0.20	5 – Low Cost 1 – High Cost
Open-source	Open -source nature of the technology allows programmers and developers to leverage other's work and customize it one's need.	0.15	5 – Openness 1 – Proprietary
Compatibility with different OS	The container tools and its manager need to be compatible with different OSs as end host can have system.	0.20	5 – Highly Compatibility 1 – Low Compatibility
Community Support & Documentation	Good community support and documentation help in easy bug fixes and in deploying technology features. A good online community provides almost 24x7 support for most issues.	0.15	5 – Superior Support 1 – No Support
Learning Curve	Technology should be easy to learn and implement.	0.10	5 – Lucid 1 – Complex
Ease of Deployment	Deployment of containers should be easy. In case of a failover, instantaneous deployment of new container should be possible.	0.20	5 – Easy to Deploy 1 – Cumbersome to Deploy

Table 26: Trade study Parameters, Weight and Reasoning for container technology

Parameter	Description	Weight	Scale
Cost	Cost is the most important factor. Team aims at incurring minimum cost of deployment and operation for this project.	0.20	5 – Low Cost 1 – High Cost
Open-source	Open-source nature of the technology allows programmers and developers to leverage other's work and customize it one's need.	0.15	5 – Openness 1 – Proprietary
Feature support	The parameter covers the technologies ability to support various Northbound and Southbound APIs to installing new features and applications at the push of a button.	0.10	5 – Multiple Features 1 – Few Features
Community Support & Documentation	Good community support and documentation help in easy bug fixes and in deploying technology features. A good online community provides almost 24x7 support for most issues.	0.15	5 – Superior Support 1 – No Support
Learning curve	Technology should be easy to learn and implement.	0.10	5 – Lucid 1 – Complex
Modularity	The ability to customize the given technology to suit project needs is critical. We shall be testing various scenarios to see how our network functions behave.	0.15	5 – Easy to Modulate 1 – Challenging to Modulate
Bare Metal Speed	A physical dedicated hardware to run virtualized OSs provides much better performance and faster execution. Type 1 hypervisors have better execution speed than Type 2.	0.15	5 – High Speed 1 – Low Speed

Table 27: Trade study Parameters, Weight and Reasoning for hypervisor technology

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

Table 28 Trade Study for SDN controller considerations

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

Table 29 Trade Study for packet analyzer considerations

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**

Table 30 Trade Study for database storage considerations

Both MySQL and SQLite serve well as database managers, however, the project requires a small database manager with limited features. The project also intends to be light at the application front. Considering all these factors, SQLite is chosen as the best database management technology.

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

Table 31 Trade Study for traffic generators considerations

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

Table 32 Trade Study for hypervisor considerations

Parameters	Weights	Dockers	Rocket	Solaris Containers
Cost	0.20	5	5	1
Open-source	0.15	4	3	1
Compatibility with different Operating Systems (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

Table 33 Trade Study for containers considerations

5.0 Selection of Baseline Design

To build a robust and redundant SDN infrastructure that supports provisioning, monitoring, and redundancy in case of failovers of the containers installed on the hosts, selection of the SDN controller is an important criterion. While all the controllers analyzed have a wide range of features and advantages over the other controllers, ONOS offers additional features that are crucial for the project requirement. Based on the factors considered in the trade-study analysis, ONOS shall be the best choice for the SDN controller. The team has decided to deploy ONOS as the SDN controller for the Containerized, Intelligent Network Functions on the Hosts project.

ONOS is an open-source, Java-based controller that is readily available to everyone. It is easy to learn, well documented, and has broad community support. The main advantage of ONOS is that it supports multiple southbound protocols such as OpenFlow, P4, NETCONF, and more. It has an interactive web framework that provides customization as per the requirement. All these characteristics provide ONOS a competitive edge over the other controllers.

While establishing a connection between the SDN controller and the network functions, it is also vital to choose an appropriate southbound protocol. The team has decided to choose OpenFlow as the southbound protocol. ONOS supports OpenFlow protocol versions from OpenFlow 1.0 to OpenFlow 1.3.

Additionally, the network functions shall be implemented on the hosts using containers. Choosing a suitable container is an indispensable requirement for this project. After careful analysis, the team believes that Docker shall be the best choice for containers. Docker is an open-source tool that is readily available to everyone. It can be deployed on both Linux and Windows OS, which gives it a competitive edge over the other containers. The main advantage of Docker is that it provides nearly bare-metal execution speed, which reduces the latency and increases the QoS of the network. In addition to that, it is highly scalable, well documented, and provides easy debugging. All these features make it the best container option amongst all.

It is essential to select a suitable hypervisor option for test VM to perform test cases efficiently with higher throughput and low latency. KVM is selected as the best hypervisor option and shall be deployed for Test VM in the project. KVM is an open-source virtualization technology that changes the Linux kernel into a hypervisor, which hosts any OS. KVM is straightforward to use and deploy. KVM runs applications at near-native speeds, making it much more efficient.

Test cases require critical traffic elements such as UDP, TCP, packet size, directional flow, and many more. Critical parameters, such as throughput, latency, bandwidth, delay, jitters, and more determine the container's performance. Due to these reasons, it is crucial to choose test tools which do not exert overhead on the test devices. Iperf is a lightweight tool that can perform the tests as mentioned above related to the critical parameters while utilizing minimal resources on the test device.

Finally, as per the trade study analysis, the team shall proceed with deploying ONOS as the SDN controller, Docker as the container, and OpenFlow as the southbound protocol to communicate between the SDN controller and the network functions hosted on containers. KVM is selected to host the Test VM due to the performance requirement of the project and Iperf as the traffic engineering tool to avoid overhead on the network devices. For packet analysis, Wireshark is chosen for its GUI, features, and OpenFlow protocol support. For database manager, SQLite is chosen due to the project's small-scale requirement. The team believes that with the implementation of the solutions as mentioned above, the project shall receive the best output.

In the end, though the tool selection took into consideration the design requirements specific to this project, the team has also considered the requirements related to future scope. Further research shall continue in this domain to improve the performance of the final product.

6.0 References

[1] Han, B., Gopalakrishnan, V., Ji, L., and Lee, S., "Network Function Virtualization: Challenges and Opportunities for Innovations", Feb 2015. [online]. Available: http://www.ttcenter.ir/ArticleFiles/ENARTICLE/3431.pdf

[2] Gedia, D., and Perigo, L., "Performance Evaluation of SDN-VNF in Virtual Machine and Container", Proc. IEEE Conf. Network Function Virtualization and Software Defined Networks (NFV-SDN), Nov 2018.

[3] Rouse, M., "SDN Controller (software-defined-networking controller), Dec-2018. [online] Available: https://searchnetworking.techtarget.com/definition/SDN-controller-software-defined-networking-controller

[4] "ONOS", *Wikipedia* [online] Available: https://en.wikipedia.org/wiki/ONOS

[5] Koshibe, Ayaka, "System Components", Sep-2016. Available: https://wiki.onosproject.org/display/ONOS/System+Components

[6] "OpenDayLight Project" [online] Available: https://en.wikipedia.org/wiki/OpenDaylight_Project

[7] "File: Architectural Framework.jpg". [online] Available: https://wiki.opendaylight.org/view/File:Architectural_Framework.jpg

[8] "COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK, Build SDN Agilely". [online] Available: https://osrg.github.io/ryu/

[9] Rao S, "SDN Series Part Four: Ryu, a Rich-Featured Open-source SDN Controller Supported by NTT Labs", Dec 2014. Available: https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs/

[10] "What is a Floodlight Controller", SDxCentral Staff, Sept 2014. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-floodlight-controller/

[11] Rao, S, "SDN Series Part Five: Floodlight, an OpenFlow Controller
", Jan 2015. [online] Available: https://thenewstack.io/sdn-series-part-v-floodlight/

[12] "What is Cisco Application Policy Infrastructure Controller (APIC)? Part 3
", SDx Central Staff, Sept 2014. [online] Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-cisco-application-policy-infrastructure-controller/

[13] "Cisco Application Policy Infrastructure Controller Data Sheet", Cisco Data Sheets, August 2019. [online] Available: https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/application-policy-infrastructure-controller-apic/datasheet-c78-739715.html

[14] "What is a Container? A standardized unit of software", docker. [online] Available: https://www.docker.com/resources/what-container

[15] Wong, G., W., "What's the Difference Between Containers and Virtual Machines", Jul 2016. [online] Available: https://www.electronicdesign.com/dev-tools/what-s-difference-between-containers-and-virtual-machines

[16] "Docker (Software)". [online] Available: https://en.wikipedia.org/wiki/Docker_(software)

[17] “Docker Features”, *JavaTpoint*. [online] Available: https://www.javatpoint.com/docker-features

[18] “Docker overview”, *Docker Inc.* [online] Available: https://docs.docker.com/engine/docker-overview/

[19] Purrier, J, “What is Rocket and How It's Different Than Docker”, *CenturyLink*, 07-Jan-2015. [online] Available: https://www.ctl.io/developers/blog/post/what-is-rocket-and-how-its-different-than-docker/

[20] “rkt, A security-minded, standards-based container engine: rkt architecture”, *Red Hat Inc*. [online] Available: https://coreos.com/rkt/docs/latest/devel/architecture.html

[21] “Solaris Containers”, *Wikipedia* [online]. Available: https://en.wikipedia.org/wiki/Solaris_Containers

[22] “Oracle Solaris Containers”, *Oracle* [online]. Available: https://www.oracle.com/technetwork/server-storage/solaris/containers-169727.html

[23] Aziz, S, “Solaris Containers (Zones) HowTo”, 09-Aug-2009 [online]. Available: https://saifulaziz.com/2009/08/09/solaris-containers-zones-howto/

[24] “Packet Analyzer”, *Wikipedia* [online]. Available: https://en.wikipedia.org/wiki/Packet_analyzer

[25] “Wireshark”, *Wikipedia* [online]. Available: https://en.wikipedia.org/wiki/Wireshark

[26] Alberto, C, “How to Perform Network Sniffing with Tshark”, *BTREME*, 29-Feb-2016 [online]. Available: https://linoxide.com/linux-how-to/network-sniffing-tshark/

[27] Styn, H, V, “tcpdump fu”, *Linux Journal LLC*, 19-Dec-2011 [online]. Available: https://www.linuxjournal.com/content/tcpdump-fu

[28] Rouse, M, “hypervisor”, *TechTarget*, Dec 2016 [online]. Available: https://searchservervirtualization.techtarget.com/definition/hypervisor

[29] Scsami, “File:Hyperviseur.png”, *Wikimedia*, 23-July-2011 [online]. Available: https://commons.wikimedia.org/wiki/File:Hyperviseur.png

[30] “What is Hypervisor and what types of hypervisors are there”, *Avada*, 13-May-2016 [online]. Available: https://vapour-apps.com/what-is-hypervisor/

[31] “ESXi Hardware Requirements”, *VMware Inc*, 31-May-2019 [online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.upgrade.doc/GUID-DEB8086A-306B-4239-BF76-E354679202FC.html

[32] Anne, S, "KVM VIRTUALIZATION: FIND HARDWARE SUPPORT KVM/VMWARE ESXI OR NOT?", 07-Feb-2013 [online]. Available: https://www.linuxnix.com/kvm-virtualization-how-to-check-my-hardware-support-kvm/

[33] Siron, E, "Hyper-V's Actual Hardware Requirements", April-2013 [online]. Available: https://www.altaro.com/hyper-v/hyper-vs-actual-hardware-requirements/

[34] "Workstation Pro FAQs", *VMware Inc* [online]. Available: https://www.vmware.com/products/workstation-pro/faqs.html

[35] Wallen, J, "VirtualBox: A cheat sheet", *CBS Interactive*, 16-Oct-2017 [online]. Available: https://www.techrepublic.com/article/virtualbox-everything-the-pros-need-to-know/

[36] "QEMU", *Wikipedia* [online]. Available: https://en.wikipedia.org/wiki/QEMU

[37] "System Requirements" [online]. Available: https://docs.oracle.com/cd/E24191_01/common/install/system_requirements.html

[38] "System Requirements", *Oracle Corporation* [online]. Available: https://dev.mysql.com/doc/mysql-monitor/8.0/en/system-prereqs-reference.html

[39] "System Requirements", *CData Software Inc* [online]. Available: http://cdn.cdata.com/help/DEA/rsb/pg_startrequirementsrsb.htm

[40] Zhang, J., et al "A survey of network traffic generation", Oct 2015 [online]. Available: https://search-proquest-com.colorado.idm.oclc.org/docview/1921147962?pq-origsite=summon

[41] Cox, J., "Best Network Traffic Generator Software & Tools for WAN and LAN Bandwidth Testing", July 2019 [online]. Available: https://www.ittsystems.com/network-traffic-generator/

[42] Stuart, B., "Testing Network Performance and Throughput with Iperf", Nov 2011 [online]. Available: https://blog.nexcess.net/testing-network-performance-and-throughput-with-iperf/

[43] Anand, D., Narasimhakumar, H., Ninale, S., Kulkarni, R., "SDN/NFV service chaining", ITP Capstone Project, May 2019