University of Colorado
Interdisciplinary Telecom Program
TLEN 5700

Concept Definition Document

SDN/NFV VNF Service Chaining

Information

Approvals

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

Project Customers

Name: Brooke Mouland Address: 1225 17th St #1690, Denver Email: bmouland@equinix.com	Name: Dr. Levi Perigo Address: ECOT 312, CU Boulder, Boulder Email: levi.perigo@colorado.edu

Team Members

Name: Dashmeet Singh Anand Email: dashmeet.anand@colorado.edu	Name: Hariharakumar Narasimhakumar Email: hariharakumar.narasimhakumar@colorado.edu
Name: Sarang Ninale Email: sarang.ninale@colorado.edu	Name: Rohit Dilip Kulkarni Email: rohit.d.kulkarni@colorado.edu

1.0 Project Description

Service Chaining or Service Function Chaining (SFC) is a capability that uses Software Defined Networking (SDN) to define a chain of network services (such as Firewall, NAT, QoS and many more) [1]. These network services are decoupled from traditional networking hardware and written in software (known as Virtual Network Functions - VNFs) to support a fully virtualized infrastructure. SDN can be used to detect traffic flows and apply different service chains based on the type of traffic. Even though the ability to connect these network services is well known, there is not enough data on the performance characteristics of chaining multiple VNFs.

Equinix has had a prominent footprint in providing Data-center Solutions and Internet Exchanges between leading networks through their availability. This “SDN/NFV VNF Service Chaining” project aims at testing the performance characteristics of different service chains of both vendor and open source VNFs. We will also be testing the most optimal order of linking these VNFs together and any losses incurred in adding a VNF to the chain. This project shall stress different service scenarios, which Equinix houses, to fully understand their virtualization capabilities and provide catalogs for their customers.

Level 1: Create service chains with Virtual Network Functions (VNFs) from various vendors or open source network services. Create specific and well-defined test cases to test the service chains for throughput and performance. The service chains will be subjected to traffic with varied parameters such as packet size, TCP, UDP, unidirectional, bidirectional traffic, number of VNFs in the service chain and many more.

Level 2: In addition to Level 1 criteria, various service chain combinations will be tested in a consistent environment. Throughput and performance parameters like CPU utilization will be evaluated for various combinations. Create an abstraction layer to plug-in and test various combinations of service chains with only the knowledge of inputs, outputs and functionality.

Level 3: In addition to Level 2 criteria, data extracted from test cases can be used for performance monitoring of the service chain. An open source tool can be used to query, alert and create a visual representation of the data stored in the database. Once the data is available, libraries will be used for packet processing to gain logical results.

Multiple VNFs running on hypervisors will be linked to form a service chain. These hypervisors will be hosted on x86 architecture devices, as depicted in the Functional Block Diagram shown in Figure 1. A testing tool will be used to generate traffic with various parameters. This traffic will be input to the service chain and the test results will be stored in databases (see Concept of Operations diagram in Figure 2). Additionally, data analysis tools will be used to provide insights into the performance characteristics of various service chain combinations.

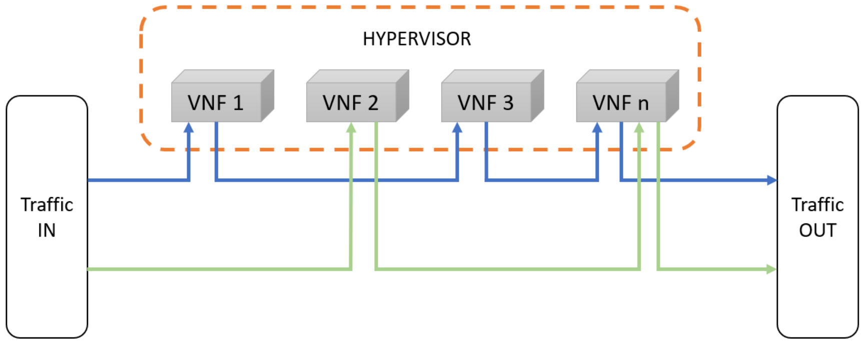


Figure 1: Functional Block Diagram (FBD)

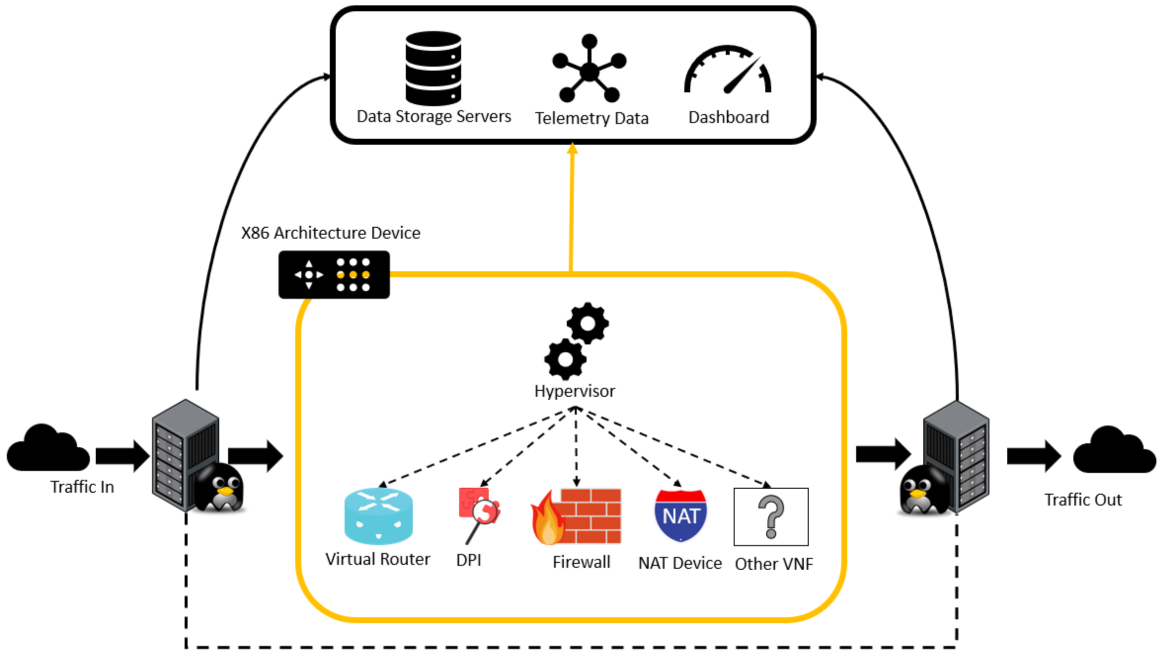


Figure 2: Concept of Operations (CONOPS)

Technical:

CPE.1.1: Create multiple service chain combinations with VNFs implemented in VMs run on one or multiple hypervisors.

CPE.1.2: Build a consistent test environment and identify test cases. The test environment should remain constant with respect to the order of VNFs and test parameters. This ensures that the test results can be reproduced for a particular combination of VNFs.

CPE.1.3: Save test results in a database and perform data analysis for evaluating performance. The telemetry data generated by the testing tool is sent to a database. Performance monitoring will be done by querying the database and will be used to create a dashboard.

Logistical:

CPE.2.1: Equinix to provide VNF images, management platform, hypervisor and access to their User Interface (UI) and Application Programmable Interface (API).

CPE.2.2: Gain extensive knowledge of Service chaining, VNFs and Software-Defined Networking to create effective service chains. Get familiar with a testing tool to effectively test the performance and throughput of service chains.

2.0 Design Requirements

Functional requirements (FNC.X) as well as design requirements (DES.X.X) for SDN/NFV VNF Service Chaining are specified below to improve project clarity.

FNC.1	This project will create multiple combinations of service chains.		
	DES.1.1	Virtual Network Functions (VNFs) will make up individual components of the service chain.	
		DES.1.1.1	Software images of VNFs for the service chain will be provisioned by Equinix.
	DES.1.2	VNFs will be hosted within hypervisors.	
		DES.1.2.1	Virtual network interfaces will be used to connect the VNFs together, forming the service chain.
	DES.1.3	The hypervisors will be hosted on x86 architecture servers.	

FNC.2	The performance of service chains will be tested.		
	DES.2.1	Well-defined, robust, and reproduceable test cases should be constructed.	
	DES.2.2	Service chains will be subjected to traffic generated by the traffic generator tools.	
		DES.2.2.1	Testing will be performed with traffic of varying parameters such as packet size, TCP, UDP, direction of traffic etc.
		DES.2.2.2	Performance parameters such as throughput, latency, CPU utilization, etc. will be measured for each test case.
FNC.3	Data obtained from the tests conducted in FNC.2 will be stored and processed.		
	DES.3.1	Results of the tests will be stored in a database.	
	DES.3.2	Gathered data will be analyzed to provide insight into the performance of the service chain in the form of a dashboard.	

Table 1: Functional and Design requirements table

3.0 Key Design Options Considered

To implement SDN/NFV VNF Service Chaining, various design options were considered with respect to hypervisors and traffic generators. The below analysis encompasses both vendor and open source alternatives available in the market that satisfy our requirements.

Hypervisors

Virtualization enables efficient management of finite hardware resources resulting in the optimized performance of servers. To administer the allocation of server hardware resources such as memory, bandwidth, and disk storage space to virtual machines, hypervisors are deployed. Hypervisors provide a platform to interact with hardware resources and environments to run multiple virtual systems, thereby increasing compatibility and efficiency.

Any change in the underlying hardware can influence the output of the project significantly. Hypervisors take control of the resources and maintain uniformity in the allocation of appropriate resources to the Virtual Network Functions (VNFs). This will help achieve the most accurate results as the test cases will not be affected by the hardware variables.

As illustrated in Figure 3, there are two types of hypervisors [3]:

Type 1: Directly interacts with system hardware.

Type 2: Requires a host Operating System, which utilizes its virtualization, networking, memory, and other resource capabilities.

While Type 1 is faster and produces better throughput, Type 2 provides better control from the Operating System (OS), as illustrated in Figure 3. The advancement in virtualization techniques in today's OS further reduces the latency that we might encounter [3]. Therefore, Type 2 Hypervisors were selected for this project.

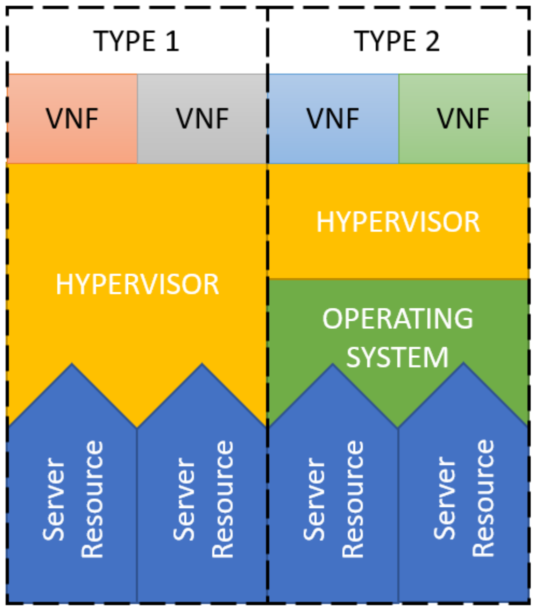


Figure 3: Type 1 and Type 2 Hypervisors [3]

VMware's ESXi

Elastic Sky X (ESXi) system is developed by VMware and is popular for its resource management capabilities. ESXi has its own kernel module known as VMKernel, capable of interacting with or hosting almost all Operating Systems [4]. The Console Operating System (COS) by VMware provides tools for executing and monitoring ESXi systems [2]. Being designed for data center and cloud services, it can handle multiple virtual machines. The functioning and support are well documented, and steps for deployment are easy. It supports x86 architecture and the majority of hardware from a wide range of vendors. License purchase is required for support and usage.

Minimum Requirements version 6.5 [5]:

- CPU needs to support: 64-bit x86 architecture.
- Host machines require two cores.
- Minimum RAM 4GB (8GB Recommended).
- Require Gigabit or Fast Ethernet Controller.
- SCSI/Local/Non-Network/RAID LUN – unpartitioned space for virtual machines.
- For SATA – SAS controller required.

Pros	Cons
Optimized resource usage.	License Purchase required.
User friendly and easier to deploy.	Complex design results in unpredictable throughput.
Guest OS support: Linux Windows Unix.	
Supports x86 architecture.	
Highly reliable and stable system.	
Secure and adheres to industrial standards.	

Table 2: Pros and cons for VMWare's ESXi

KVM

Kernel-based Virtual Machine is an open sourced Linux based hypervisor control. KVM supports multiple architectures and it continues to develop for a wide range of products. KVM comes native to Linux or Ubuntu systems with x86 and x64 architectures. It uses threads to run virtual machines, thereby utilizing the host machine's kernel processes [6]. KVM uses QEMU to manage and emulate resources from the host [2]. It is a lightweight system and requires minimum hardware resources for operations.

Minimum Requirements [6]:

- Single core and should be able to thread for each virtualized machine.
- Minimum RAM of 2GB.
- 6GB of disk space.
- Supports both 32 and 64-bit architecture.
- Needs Linux host machine if running as Type 2 Hypervisor.

Pros	Cons
Linux-based system.	Host machine should be Linux.
Open Source.	Throughput is dependent on host kernel.
Widely deployed and good support.	
Natively present in most LINUX systems.	
Guest OS support: Linux Windows Unix.	

Table 3: Pros and cons for KVM

XEN

Developed by the University of Cambridge, XEN is a software-based hypervisor [2]. Although it is open source, it is majorly used and developed by the Citrix systems. Xen has better control over the system due to the exclusive administration control it achieves over the host machine [2].

Minimum Requirements [8]:

- 64-bit x86 CPUs.
- Minimum RAM of 2GB (4GB Recommended).
- Locally attached storage with 100MB of storage.
- 100Mbit/s and faster NIC and IPv4 capabilities.

Pros	Cons
Open source project but managed by Citrix system.	Limited host and guest systems support.
Better throughput from host kernel.	
Supports x86 architecture.	

Table 4: Pros and cons for XEN

Hyper-V

Hyper-V comes native to Windows Server Operating Systems. It supports multiple virtual machines over one single server [9]. Hyper-V requires a Windows Operating System and uses the host's kernel for interaction with the hardware [2]. Hyper-V first emulates the hardware over a parent thread and then subdivides them according to the client system's requirement. This prevents direct connectivity of the client systems to the hardware. Hyper-V runs the client machines over independent child partitions, thereby separating the resources [2]. Hyper-V is a licensed software with support from Microsoft team.

Minimum Requirement [9]:

- Host machine should have Windows Server 2008 or Windows 8.
- CPU should be x64 architecture.
- Minimum RAM of 4GB.
- 1GHz of Processor with at least 2 Cores.

Pros	Cons
Good support.	License purchase required.
Fast and stable systems.	Host only Windows OS.
Have built-in security features.	Limited network functionalities.
Supports x86 architecture.	

Table 5: Pros and cons for Hyper-V

Traffic generators

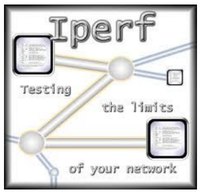
Traffic generators are tools used to generate and monitor traffic across the network to be tested. These tools are capable of emulating client/server interactions such as bursts of traffic and keepalives. While the generated traffic traverses the network, performance parameters such as throughput, latency, congestion etc. can be monitored for evaluation.

Network traffic generation can be classified into:

- Traffic generation based on network traffic model: Generated traffic conforms to the characteristics of the traffic model.
- Traffic generation based on traffic characteristics: Can be either per-flow level or per-packet level.
  A flow is defined as a collection of traffic from a source to a destination. Actions are specified for the packets that match a flow. This type of traffic generation focuses on the end-points.
  Packet-level traffic generation can be used to modify parameters such as packet size. Such types of traffic focus on individual networks and the ability of the network to handle the extreme traffic load.
- Traffic generation based on application protocol: Traffic is generated considering the application protocol to be tested. For instance, iPerf can test bandwidth for TCP and UDP traffic. [10]

iPerf

iPerf is a tool for measuring TCP and UDP benchmark for a network. As shaping algorithms have been implemented, it is suitable for testing UDP traffic. The iPerf toolkit consists of a combined client/server program. All server output remains on the server end and is not sent back to the client. Also, only one socket for data will be opened as compared to two for Netperf [11][15].



Pros	Cons
Relatively simpler to implement.	Cannot monitor CPU usage.
“pthread” library allows multithreading to generate multiple data streams between source and destination hosts.	Does not support protocols apart from TCP and UDP.
Supports both IPv4 and IPv6 traffic.	With multiple streams, dedicated scripts are necessary to connect the output from the clients with servers.
The results of the test can be saved in a log file.	

Table 6: Pros and cons for iPerf

Netperf

Developed by Hewlett-Packard, Netperf is used to test network performance for both TCP and UDP traffic. Since no shaping algorithms have been implemented in Netperf, the nature of UDP tests are limited as the sender might overflow the receiver. Netperf requires a client and server instance to be present on the end devices. Two connections are required between the server and the client; a communication socket for control information and a data socket for sending test data [11].

Pros	Cons
The two distinct connections allow the server to be completely controlled by the client.	Limited UDP tests possible.
Can be used to calculate CPU usage of the devices in the network.	Does not support creation of log files.
Supports both IPv4 and IPv6 traffic.	Not possible to specify the destination port for data sockets.
	Single threaded and requires multiple instances to generate simultaneous tests, resulting in consumption of system resources.
	Does not support protocols apart from TCP and UDP.

Table 7: Pros and cons for Netperf

Ixia

Ixia provides hardware test generators which can be used to test network performance for a wide range of network types. Ixia has made its traffic generators to perform benchmark tests as recommended in RFC 2544, which is the standard for testing network devices. Ixia provides a Graphical User Interface (GUI) - IxExplorer for managing Ixia test hardware [13]. The GUI provides complete control over generation and analysis of Layer 2-Layer 4 traffic streams on a wide range of interface technologies, including Ethernet, Gigabit and 10 Gigabit Ethernet, Packet over SONET (POS), ATM, Frame Relay etc [12][15].

Pros	Cons
Ixia provides a wide range of chassis options with varying port densities to fit the needs of the network.	Ixia manufactures hardware test generators compared to their open source software counterparts which do not require a separate equipment for testing.
Test cases generated by Ixia can be automated.	Cost of the hardware test generators are very high.
Includes analysis tools to provide insight into the network performance instead of just providing statistics of the tests.	Steep learning curve.
The GUI provides quick and easy configuration of multiple parameters at once, which allows management of large-scale tests.	Poor documentation.
A wide range of filters are available to test multiple protocols such as TCP, UDP, IPv4, IPv6, ARP, DHCP, OSPF and many more.	

Table 8: Pros and cons for Ixia

Spirent TestCenter

Spirent TestCenter is an end-to-end network performance testing solution. The user interface is designed to improve the tester's efficiency by providing customization options to meet the unique needs of testers [14]. It can be used in varying network types including service providers, Network Equipment Manufacturers (NEMs) and enterprises.

Pros	Cons
The GUI is intuitive and easy to use.	Requires the use of hardware appliances for generating test traffic.
Allows automation of test cases involving repetitive tasks.	Even though Spirent TestCenter can be rented for a nominal price compared to other commercial options like Ixia, the cost incurred is high in the long run compared to open source test generators.
Can emulate a wide range of traffic types - including voice, video and data traffic.	
Built-in plugins and configuration accelerators reduce the time to test.	
Object oriented interface simplifies the creation of multi-protocol topologies.	

Table 9: Pros and cons for Spirent TestCenter

4.0 Trade Study Process and Results

The evaluation of multiple available options is a good practice to make an informed decision. Table 10 describes some parameters based on which components were analyzed. Each parameter in the table is associated with a weight that signifies the importance of the parameter with respect to the project design. Hypervisors and traffic monitoring tools were evaluated individually in Table 11 and Table 12 respectively against the parameters mentioned in Table 10.

Parameter	Description	Weight	Scale
Open source	An open source software has its source code available freely. This gives programmers more control, as they can inspect the code and alter it according to their needs.	0.1	5 - High level of control. 1 – Low control.
Cost	An important aspect of a project is the cost incurred in its development and operation. During the course of the project, the team aims to minimize the budget, without compromising quality.	0.2	5 – Low cost. 1 – High cost.
Industrial applications	Software that are widely accepted and deployed in the industry are stable and well documented. The tools should adhere to industry standard requirements.	0.15	5 – High popularity. 1 – Less popularity.
Support	A software with good support is backed by frequent bug fixes and enhancements from the developers. The team may face difficulties during the implementation of various	0.10	5 – Superior support from developers.

	combinations of service chains and their testing. This parameter characterizes the technical support provided by the online support community and forums.		1 – No support from developers.
Compatibility with Operating Systems	A tool that can be used on multiple Operating Systems (OS) gives an engineer the flexibility to test and deploy a product in various environments. The software should be independent of the underlying platforms that support it.	0.20	5 – High compatibility. 1 – Low compatibility.
Performance	This parameter will be used to analyze the impact of the tools on the working of the project. The project must have components which will take up minimal resources without affecting the output.	0.25	5 – Least impact on hardware performance. 1 – Highest impact on hardware performance.

Table 10: Trade study Parameters, Weights and Reasoning

Parameter	Weights	VMWare ESXi	KVM	XEN	Hyper-V
Open source	0.1	1	**5**	3	1
Cost	0.2	3	**5**	5	2
Industrial applications	0.15	4	**4**	2	3
Support	0.1	5	**4**	3	5
Compatibility with Operating Systems	0.2	5	**3**	3	3
Performance	0.25	5	**5**	3	4
Total	1	4.05	**4.35**	3.25	3.05

Table 11: Trade study for hypervisor considerations

Parameter	Weights	Netperf	iPerf	Ixia	Spirent
Open source	0.1	5	**5**	1	1
Cost	0.2	5	**5**	1	1
Industrial applications	0.15	2	**3**	4	4
Support	0.1	5	**5**	4	4
Compatibility with Operating Systems	0.2	4	**4**	3	4
Performance	0.25	3	**4**	5	5
Total	1	3.85	**4.25**	3.15	3.1

Table 12: Trade study for test generator considerations

5.0 Selection of Baseline

After careful consideration and evaluation of available options, the selection comes down to KVM hypervisor for managing Virtual Network Functions and iPerf for measuring performance parameters.

The project requires analysis of critical parameters like bandwidth, throughput, and memory utilization in the virtual environment. Since the project is performance-centric, it becomes imperative to use tools that do not exert overhead on the hardware. iPerf is a lightweight tool that can measure the throughput of the network using minimal resources making it suitable for our project.

The major driving factor in selecting KVM was its popularity in the networking industry. Many tech giants including Equinix manage their virtualized data center environment using KVM. This would help in building a test environment equitable to Equinix's infrastructure.

KVM and iPerf, being popular open source tools, are widely deployed and are serving many industry critical applications. Both the tools have a strong presence and support of the developer community that could be beneficial in seeking guidance and information in case of any roadblocks during the project.

IPerf and KVM are compatible with most Linux based Operating Systems (OS) and processing units that are widely used today. Additionally, KVM supports the use of different guest OS like Windows, Linux, and Unix. This allows flexibility and ease in deploying the servers which form the foundation of the testing environment.

Cost plays a key role in the development of any system. It is necessary to provide an efficient solution for the least possible capital. The selection criteria have prioritized use of components that are available in the market at no costs. Our baseline applications KVM and iPerf are equivalent to the proprietary software in terms of performance and results, thereby fulfilling most requirements for the project.

All the above-mentioned factors have led to the selection of KVM hypervisor and iPerf with the trade study scores of 4.35 and 4.25 as per the analysis in Table 11 and Table 12 respectively.

With the selection of the above hypervisor and performance monitoring tool, the team will move forward to implement this idea. Although the scope of this document is limited to current customer requirements and our knowledge to this point, the team has considered the possibility of changes in requirements from the customer in the future. Further research shall thus continue towards improvement in the form of both services and tools. However, changes will be made if the research results in significant benefits over the current considerations. This approach allows incorporation of better features during the project lifecycle and ensures fulfillment of customer requirements with the highest standards.

6.0 References

[1] B. Carpenter, "Middleboxes: Taxonomy and Issues", *Ietf.org*, 2002, [Online]. Available: https://tools.ietf.org/html/rfc3234. [Accessed: 24- Sep- 2018]

[2] Desai, R. Oza, P. Sharma, and B. Patel, "Hypervisor: A Survey on Concepts and Taxonomy", *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2013. [Online]. Available: https://pdfs.semanticscholar.org/650f/e84a42c2914fb94c282cb3736c48e506d462.pdf [Accessed: 23- Oct- 2018]

[3] "Learn about hypervisors, system virtualization, and how it works in a cloud environment," Hypervisors, virtualization, and the cloud, 23-Sep-2011. [Online]. Available: https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/index.html. [Accessed: 24-Oct-2018].

[4] "ESXi | Bare Metal Hypervisor," *VMWare*, 25-Sep-2018. [Online]. Available: https://www.vmware.com/products/esxi-and-esx.html. [Accessed: 24-Oct-2018].

[5] "vSphere Installation and Setup", *VMware, Inc.* [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.5/vsphere-esxi-vcenter-server-65-installation-setup-guide.pdf [Accessed: 23- Oct- 2018]

[6] "Red Hat Customer Portal," *Chapter 1. System Requirements.* [Online]. Available: https://access.redhat.com/documentation/en-

us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/chap-requirements#sect-host_requirements. [Accessed: 24-Oct-2018].

[7] “vSphere Installation and Setup”, *VMware, Inc.* [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.5/vsphere-esxi-vcenter-server-65-installation-setup-guide.pdf [Accessed: 23- Oct- 2018]

[8] System requirements. [Online]. Available: https://docs.citrix.com/en-us/xenserver/current-release/system-requirements.html. [Accessed: 24-Oct-2018].

[9] J. Vigo, “Microsoft Hyper-V: The smart person's guide,” *TechRepublic*. [Online]. Available: https://www.techrepublic.com/article/microsoft-hyper-v-the-smart-persons-guide/. [Accessed: 24-Oct-2018].

[10] J. Zhang et al, "A survey of network traffic generation," 2015, Available: https://colorado.idm.oclc.org/login?url=https://search-proquest-com.colorado.idm.oclc.org/docview/1776483150?accountid=14503.

[11] Network Test Tools. [Online]. Available: https://staff.science.uva.nl/j.blom/gigaport/tools/test_tools.html. [Accessed: 24-Oct-2018].

[12] N. Lekhak, “Networking Fundamentals and Certification Blog,” *Ixia Traffic Generator*, 01-Jan-1970. [Online]. Available: https://lekhaknawraj.blogspot.com/2012/12/ixia-traffic-generator.html. [Accessed: 24-Oct-2018].

[13] A. Chaudhary, “Testing Solution for IP Networks, Ixia-IxExplorer” *LinkedIn SlideShare*, 13-Jun-2013. [Online]. Available: https://www.slideshare.net/nlekh/ixiaexplorer. [Accessed: 24-Oct-2018].

[14] “Spirent TestCenter Software,” *Spirent TestCenter Software - Spirent*. [Online]. Available: https://www.spirent.com/Products/TestCenter/Platforms/Software. [Accessed: 24-Oct-2018].

[15] S. S. Kolahi, S. Narayan, D. D. T. Nguyen and Y. Sunarto "Performance Monitoring of Various Network Traffic Generators", *Unitec New Zealand*. [Online]. Available: https://unitec.researchbank.ac.nz/bitstream/handle/10652/3699/Performance%20Monitoring%20of%20Various%20Network%20Traffic%20Generator