Segment Routing Green Spine Switch Management

Systems for DCN

By

Ose Osamudiamen

A thesis submitted to the Faculty of Graduate and Postdoctoral
Affairs in partial fulfillment of the requirements for the degree of

Masters of Applied Science

in

Systems and Computer Engineering

Carleton University
Ottawa, Ontario

© 2017, Ose Osamudiamen

Abstract

It is without a doubt that the data center is the core of most IT industries, providing services to billions of users today. The preferred data center network (DCN) architecture used for this platform is the 2-tier leaf-spine topology, which is not only energy efficient but has a good performance metrics. Furthermore, due to the exponential increase of information and services from datacenters, it is projected that a 14.1 zeta bits of bandwidth would be needed to meet the current demands of data by the end of 2020 in USA data centers alone. This would require a tremendous amount of energy to run those data centers. Hence, in recent years there is more focus on reducing the energy consumption in a data center. In trying to reduce the energy consumption of a data center, this thesis approaches the problem by utilizing Segment Routing-Traffic Engineering, and other power algorithms for an efficient DCN. The proposed approach makes it possible to deactivate spine switches and links on the data center networks, which results in energy savings. Our experimental results yielded an approximate of 80% in energy savings of the spine switches consumption whilst maintaining an average similar traffic performance.

Acknowledgements

First, and foremost, I would like to acknowledge, my supervisor Prof Chung-Horng Lung of Carleton University, for giving the opportunity to carry out this thesis work. He became not only an academic supervisor, but a resourceful help, and beacon of hope to complete my research work.

I want to acknowledge Mr Busuyi, he became an uncle where I needed one, an elderly friend. His constant help are still invaluable to me.

I want to recognize my very good friend Adegoke Falaiye, who assisted me when I was in the mire. Without his help I do not think I would a chance nor opportunity to complete this task.

I also want to recognize all those who assisted me in a big and small way. Because, when I was down, their assistance proved very helpful to me, this includes, Onwuka Oke, Uchechukwu, Dasola Oluge, Adegoke Oluwasegun, Gboyega Onagbolag, Mr Femi, Mr Elom, Tola Adegbola, Idowu Emmanuel, Greg, Lee, Bryan, Zheng, Hui, and Simi.

Also important, I want to acknowledge my brothers Eghosa, Osagie and sister Ikponmwosa, and family members here in Ontario. They were of good support and pillar to me, with good advice and words of encouragement, that kept me going.

I give thanks, and gratitude to my parents for everything that makes me to be me.

My ultimate thanks, goes to my creator Jehovah, he has shown in so many ways, that he cares for me, when I am down, he strengthens me. For when I have falling down he picks me up. He gave me the strength, wisdom and fortitude to complete this research work. I am nothing without him.

Table of Contents

Abstract.....	ii
Acknowledgements	iii
Table of Contents	v
List of Tables	viii
List of Figures.....	viii
List of Acronyms	v
List of Appendices.....	xiii
Chapter 1: An Introduction	1
1.1 Motivation	2
1.2 Energy Distribution in DCNs	3
1.3 Main Contribution of this thesis	6
1.4 Thesis Organization	8
Chapter 2: Background and Related works	9
2.1 Related works.....	11
2.2 Background	12
2.2.1 Virtualization in Tierd DCN.....	12
2.2.1.1 Energy Efficiency in DCN using virtualization	14
2.2.1.2 Energy Efficiency in DCN Communication system	15
2.2.2 DCN Communication infrastructures	16
2.2.2.1 DCN switch-centric topologies	17
2.2.2.1.2 Fat-tree Topology	18
2.2.2.1.3 Diamond Topology	20
2.2.2.1.4 Leaf and Spine Topology	21
2.2.2.1.5 F ² Tree Topology	22

2.3	Comparing the leaf-spine over some switch-centric topology in a DCN	23
2.4	Software Defined Networking (SDN)	28
2.5	Introduction to Traffic Engineering	32
2.5.1	Traffic Engineering - (MPLS)	31
2.5.2	MPLS-TE in SDN Networks	35
2.5.3	Benefits of SDN-TE over MPLS-TE	37
2.5.4	Traffic Engineering using auto-bandwidth	37
2.5.5	Traffic Engineering using QoS	40
2.5.6	Traffic Engineering using ECMP and UNECMP	41
2.6	Introduction to segment routing	41
2.6.1	Segment Routing.....	42
2.6.2	Types of Segments	45
2.6.3	Segment Routing in a leaf-spine DCN.....	48
Chapter 3: Segment Routing Green Spine Switch Management Systems for DCNs		
.....		50
3.1	The SR-GSSMS.....	50
3.1.1	Important highlights of the SR-GSSMS	51
3.2	The SR-GSSMS Modules	55
3.2.1	The Segment routing & routing module	58
3.3	Algorithms	66
3.3.1	Algorithm 1: Segment routing with auto-bandwidth	66
3.3.2	The Power control module and other SR algorithms	69
3.3.2.1	Algorithm 2: Decreasing the active spine switches.....	71
3.3.2.2	Algorithm 3: Decreasing the active spine links	72
3.3.2.3	Algorithm 4: Increasing the active spine switches	73

3.3.2.4 Algorithm 5: Increasing the active spine links	74
Chapter 4: Simulation Setup and Results	76
4.1 Energy Savings with SR-GSSMS in DCN	79
4.2 Comparing SR-GSSMS to default SDN System	82
4.2.1 Comparing PLP on SR-GSSMS to default routing System	82
4.2.2 Performance Measure with varying Packet sizes	84
4.2.3 Time taken to Connect Network to Controller.	87
4.2.4 Measure of the Latency	88
4.3 The effect of varying some switch metrics on the networks performance	89
Chapter 5: Conclusion and Future works	93
5.1 Future works	93
References	95
Appendices.....	106
A.1 Segment routing auto-bandwidth in action	108

List of Tables

Table 2.1	Showing comparison of 3 DCN	24
Table 2.2	Cost and energy efficiency	25
Table 2.3	Performance metrics with increase in the number of servers	25
Table 3.1	The forwarding table for node 1	61
Table 3.2	The new forwarding table for node 1	62
Table 3.3	List of Notations	66
Table 4.1	List of experimental parameters and values used	78

List of Figures

Figure 2.1 Cisco massively scalable DCN	14
Figure 2.2 A simple Google Fat-tree	19
Figure 2.3 A diamond DCN topology improved from Fat-tree	21
Figure 2.4 A simple Facebook leaf-spine topology	22
Figure 2.5 A F ² Tree DCN Topology	23
Figure 2.6 The SDN architecture with API's	29
Figure 2.7 The SDN infrastructure	32
Figure 2.8 Traditional network IP/MPLS vs SDN MPLS	37
Figure 2.9 Adjacency Segment Routing	47
Figure 2.10 Prefix Segment Routing	48
Figure 3.1 Green energy management system	51
Figure 3.2 Network state highlighting backup links with SR	53
Figure 3.3 Showing ECMP with default routing in DCN.....	60
Figure 3.4 Modelled nodal-SR on leaf-spine DCN	62
Figure 3.5 Modelled-SR policy with auto-bandwidth	64
Figure 3.6 Active and inactive switches in action	70
Figure 4.1 POPC performance results	80
Figure 4.2 PLP performance with changing ANASS	81
Figure 4.3 Comparing PLP with and without SR	83
Figure 4.4 Comparing PLP of network compared to different UDP transmitted	86
Figure 4.5 Time taken to Connect links to Controller	87
Figure 4.6 Measure of Latency across the network	89

Figure 4.7 The effect of varying SuT-H	90
Figure 4.8 The effect of varying SuT-L	91

List of Acronyms

APIs	Application Programming Interfaces
ANASS	Active Number of Active Spine Switches
BW	Bandwidth
CISC	Complex Instruction Set Computer
DCN	Data Center Network
DDR3	Double Data Rate RAM type 3
DRAM	Dynamic Random Access Memory
ECMP	Equal Cost Multiple Path
ECLB	Equal Cost Load Balancing
FEC	Forwarding Equivalence Class
GSSMS	Green Spine Switch Management System
HPC	High Performance Computing
IaaS	Infrastructure as a Service
IOPS	Input Output Operations per Seconds
ISP	Internet Service Providers
LSPs	Label-Switched-Paths
MPLS	Multi Protocol Label Switching
ODL	Open daylight
PaaS	Platform as a Service
POPC	Percentage of Power Consumed
PHB	Per Hop Behavior

PLP	Percentage of Lost Packets
QoS	Quality of Service
RISC	Reduced instruction set computer
RSVP	Resource Reservation Protocol
SaaS	Software as a Service
SD-WAN	Software Defined – Wide Area Network
SoC	System on Chips
SPF	Shortest Path Forwarding
SPRING	Source Packet Routing in Networking
SR-GSSMS	Segment Routing-Green Spine Switch Management System
TE	Traffic Engineering
TE-IGP	Traffic Engineering-Interior Gateway Protocol
UCLB	Unequal Cost Load balancing
UNECMP	Un-Equal Cost Multiple Path

List of Appendices

A.1	Segment routing auto-bandwidth in action	88
------------	---	-----------

Chapter 1: Introduction

The data center is the core infrastructure for most IT architectures. More and more business applications, software, and platforms are hosted in a Data Center Networks (DCN). Furthermore, a data center can be used to provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), as provided by Amazon Web Services [2] [97], Google-cloud services [98], and Windows Azure services [99] infrastructure, just to name a few which is the current trend for most IT business and new start-up IT companies. Hence, the network architecture of a data center is pivotal to its performance, resilience, elasticity, and scalability. Added to these, the data center must be capable of quickly deploying and supporting added services almost instantaneously. With these requirements in the background, the network design of most datacenter is pivotal to creating avenues for scalability, flexibility, performance, resilience, maintenance, and energy efficiency.

There is a continuous exponential increase in demand of bandwidth used in data centers. This results in an expansion of many datacenters, resulting in a very high energy consumption. To reduce the energy consumed in data centers as well as to meet the high demand of designing data centers, in terms of network performance, Traffic Engineering (TE) has been considered and deployed. TE is simply changing the direction of a flow from one point to another, in other to improve the performance on a network. TE has evolved over the years, to meet the basic requirements of building an efficient DCN. Recently, segment routing a form of traffic engineering has been developed. The segment routing technology, provides the platform of integrating numerous networking functions to

enhance the performance of DCNs, it is much easier to deploy and it also makes it better to integrate traditional network to Software Defined Networking (SDN) infrastructure. By taking advantage of the main SDN technology, i.e., the separation of the data-plane and control-plane in a network, and combining traffic engineering with segment routing, as well as proposed algorithms for power consumptions for a DCNs, this thesis has been able to attain, a high energy efficient system for DCNs.

1.1 Motivation

There is much emphasis on energy efficiency in a data centers. This is because a standard datacenter consumes a significant amount of energy. For example, in 2014, data centers in the U.S. alone, consumed 70 billion KW of electricity [48]. Furthermore, an extrapolation of the Internet bandwidth consumption as projected by Cisco will reach an increase of 14.1 zettabytes of bandwidth by 2020 [49]. One can conclude that, when combining these potential demands; to sustain this bandwidth consumption, would demand tremendous energy to run and maintain the DCNs. Hence, in recent years, much emphasis has been placed on green energy and energy savings in a DCN.

DCN has been built over the years with various kind of topologies. The research work of [3] [5] [8] [10] [11] [83], has shown how vital the topology used in building a DCN is essential to energy savings, as well as its performance. Hence, based on these works and Cisco's recommendation [4] [91], the topology used in these thesis work, was the leaf-spine topology. Further analysis in this thesis, showed the reason for selecting the leaf-spine DCNs. In [1], the leaf-spine topology was used to develop a green spine switch

management system (GSSMS), by deploying a couple of proposed power algorithms on the DCN, these algorithms yielded an energy savings of up to 80%, by putting some spine switches to rest while transferring the functionality to some active spine switches.

Furthermore, there are various ways that have been adopted over the years to deploy green energy on the DCN. The green energy method of interest in this thesis is TE on DCN topologies. Segment routing, is one of the most recent TE technology developed [74] [78]. It was developed to make traffic engineering easy for network engineers. Hence it has many advantages over the previously used methods of TE such as MPLS, it is also developed to make traditional network integration with SDN easier. In the thesis work of [50], segment routing was used on a 4-ary Fat-tree DCN in [50]. The power algorithm used in this work combined with segment routing yielded energy savings in the DCN, by deactivating numerous links. In this thesis, the proposed power algorithms of [1], as well as the power algorithms using segment routing of [50], was used to develop a highly energy efficient DCN, called the Segment Routing, Green Spine Switch Management System (SR-GSSM).

1.2 Energy Distribution in DCNs

The energy distribution in a DCN as a whole can be separated into 4 units [40] considering datacenters energy consumptions [20] [21] [37] [40]:

- i) The IT equipment energy, this in total is approximately 66.5%. This includes storage, servers, switches etc.
- ii) Cooling devices and the cooling systems 29%.
- iii) Poor IT architecture, and building 4%

iv) Lighting and general receptacle 1%

This statistics shows the energy consumed by IT devices has the highest energy consumption. Hence, in reducing energy consumed in a DCN, more emphasis has been placed on IT devices. If the green energy technology on IT devices in a DCN is properly deployed, it is projected there would be an estimate of 620 billion KW of energy savings, from 2010 to 2020 [48]. Thus, the emphasis of the thesis is in green energy management in a data center.

One of the pivotal phase in a DCN is deploying virtual machines (VM) in servers. A VM architecture can be said to be a simple light weight computer, providing a good platform to deploy servers, applications, networking systems on it [100] [101]. It has the added advantage of continuous mobility which a computer does not have, hence using it as servers in a DCN, has yielded high energy efficiency [100].

From the survey conducted on data center energy saving methods in [5] [40], one can infer that these methods can be divided into two major techniques; (i) energy savings using hardware such as proper ventilation systems, green energy techniques for power generation, and (ii) software methods, such as using traffic engineering. Both methods with a focus on the main resources, the cooling systems, the server, the storage, the networking devices, and networking topologies. From the surveys conducted by various researchers [1] [2] [5] [6] [40] [43] [50], it can be seen that in total the power consumed in a data center on IT devices can be categorized percentage wise into these areas [1] [5] [37] [40] [85]:

- The servers and other computing devices (VMs) resources consume in the range of ~60%. This value, when active devices scale linearly, amounts to an energy consumption of the range 52–90%.
- The network links consumes between 10–30% but mostly below 10% of the entire DCN.
- Storage account for energy consumption in the range of 50–70% while memory accounts for 5% of the DCN energy.
- The power lost 0-8%, in the poor arrangements of devices on sites.
- The power consumed in cooling onsite devices accounts for 10–48%.

From various surveys carried out on DCN topologies, we can say that there are two major architectures used, the switch-centric, and the server-centric topologies [5] [84] [85] [86] [87]. It can be summarized, that the most preferred topology used for building DCNs which integrates the use of virtualization technologies, with better network performance as to throughput and congestions, scalability, and latency is the switch-centric topology (a DCN topology such that the routing technique is centered on the switches) [83-87]. In this thesis, the clos topology (A multistage circuit switching systems used for data centers architecture) is used interchangeably for the switch-centric topology based on the original design for a clos topology used for DCN (although not all clos topologies are switch-centric).

A recent study in [83], focused on the performance of various DCN topologies in the switch-centric family, shows that for building a DCN consisting of the same number of

hosts, VMs, and leaf switches, the leaf-spine infrastructure, which is a two-tier topology, consumes less energy compared to the traditional 3-tier clos topologies, such as Fat-tree, F²-Tree, Diamond, etc. A further explanation the switch-centric topologies is provided in Section 2.2 and 2.3 of this thesis. Hence, since the core of this thesis work is central on energy efficient DCNs, the topology of preference is the leaf and spine DCN [4]. In support of this topology, for building warehouse-scale computers for data centers, Cisco has recommended the leaf-spine data center infrastructure. This topology, has evolved and has become recently popular for building many DCN [4] [91].

The emphasis of the next chapter will be focussed on to the communication networks of data centers, and what methods have been developed over the past for energy reduction on the DCNs.

It should be noted that terms such as datacenters and data centers, leaf-spine and spine-leaf are used interchangeably in this thesis write up.

1.3 Main contribution of the thesis

There have been some proposed works to reduce the energy consumed in a DCN using green techniques and traffic engineering technologies. The main contribution of this thesis is to deploy the SR-GSSMS on a leaf-spine DCN, by taking advantage of the recent traffic engineering algorithm called segment routing. Due to the flexibility of usage of segment routing, the flow of packets in the network can be controlled to maximize the usage of some network resources on the spine switches and links, keeping the unused resources

dormant or inactive. This results in reducing the energy consumed on the leaf-spine DCN as a whole.

This thesis, combines efficiently the proposed green energy technology of segment routing as used in [50], i.e., the algorithm of minimizing the links when using segment routing and auto-bandwidth (BW) in a DCN (as recommended by IEEE 802.3az [22] [50]), and the power control algorithm to reduce the number of active switches in a leaf-spine network using the Green Spine Switch Management Systems GSSMS by Xiaolin, et al [1], to form the SR-GSSMS topology.

To summarize, the main contributions of this thesis as to energy efficiency in a DCN is to:

- Proposing a proactive segment routing based approach in the SDN controller, called the SR-GSSMS, to manage the packet flow and bandwidth usage in the data-plane of the spine switches in a scalable leaf-spine network, of oversubscription ratio 1:1 in the network. (See sections 2.2 and 2.3 for further explanation of leaf-spine topology and related terms.) The approach results in higher performance with reduced packet loss and reduction of active links and spine switches.
- A thorough simulation analysis of the control of the network bandwidth, the impact of reducing the spine-switches, the spine-links as to performance as well as energy savings on the DCN.

1.4 Thesis Organization

The remaining part of this thesis is organized as follows. The next chapter focusses on related work and background leading to this research work. It further describes the infrastructure of the SDN platform used, so as to understand the network architecture used for this research work. Chapter 3 discusses methodology and all related algorithms used in this thesis. Chapter 4 focusses on simulation, results, and explanations of the results. Chapter 5 covers the conclusions and future work.

Chapter 2: Background and Related Works

In communication systems, the data center network is arguably the most difficult problem to design for a network architect-engineer. When active, just the networks links could consume more than 10% of the energy in an entire data center [40]. With recent development, ideally, a DCN should connect nodes in the network such that any pair of host would be able to communicate with each other at the line speed of the DCN connection. The network design of the data center should be easily scalable with very high performance and DCNs resources utilization. To achieve this basic requirements many proposed topologies over the years [3]. These designs and topologies will be discussed in this chapter.

Before going further in related works, it is pertinent to understand the basic concept of how the transmission rate, the channel BW of a switch and links are all related to the energy consumption of DCNs. The famous Shannon's channel capacity formulae (equation 2.1 & equation 2.2 below) [52] shows:

$$S = \frac{kE_b}{T} = RE_b \quad (2.1)$$

Where:

- E_b is the energy for transmission per bit (J) (Joules).
- k is the number of bits transmitted per second
- S is the power in watts (W)
- R is the transmission rate of the system = k/T in bits/s
- T is duration in seconds

From this fundamental communication equation for a device, it can be inferred that the energy consumption of the devices used is directly proportional to the channel bandwidth in other words transmission per bit. This can be further categorized by the internal architecture of the switches and servers, considering [23]:

- The memories, using DRAM over DDR3
- Computing systems unit
 - Dynamic voltage and frequency scaling of the computing unit [6]
 - Advanced configuration and power interface [7]
 - Reduced instruction set computer (RISC) which is more efficient over complex instruction set (CSIC) [8]
 - The Arm and x86 processors, with Arm four times more efficient to x86 processor [9]
 - An In-depth comparison of System on chip (SoC) to RISC and CSIC, shows SoC has higher energy savings, especially on performance metrics such as requests per server, and IOPS (Input Output Operations Per Seconds) which is 50% more thermal efficient and saves more spaces 300-500% space efficiency compared to RISC. [8] [9] [10].
 - A trade-off between performance per dollar of mobile servers to embedded and volume servers, with mobile servers 300% more energy efficient over embedded and volume servers. Nonetheless, they have poorer IOPS and GPUs for intensive workloads [11].
 - To optimize the performance of SoC's servers conforming to Amdahl's law: task scheduling techniques, have been introduced to SoC's

compared to Collaborative Expandable Micro-Slice (CEM) [12] [13] [14].

Hence, from the equation 2.1 as well as the points highlighted on the internal architecture of the switches and servers, the factors needed to be considered can be thus summarized: the type of switches used in the system, be it hybrid switches, servers deployed as VM in switches, the Central Processing Unit (CPU) used in the switches, the number of active ports on the switches, the working time of a switch, the power consumption of the ports of a switch, the active time of the ports on the switch. The network topology, and the number of links used in the network all play a role in the amount of energy consumed in a DCN.

Assuming all factors are the same, since the switches, the network and the links used have a high percentage of the power consumption, it is pertinent to consider the DCN infrastructure (Architecture), and how this affects the energy efficiency.

2.1 Related works

Some works have been done in the field of green energy system on a DCN. Some have been focused on optimization of the heating and cooling system, others have been on the type of CPU used [12] [13] [14], some the network topologies [5] [24] [25] [26], and some have used TE to reduce the energy consumed in the network [1] [11] [12] [32] [35] [38] [40] [50]. The methods used in [1] and [50], are of importance to the SR-GSSMS.

2.20 Background

Having highlighted the importance of parameters necessary in the choice for DCN infrastructure, we would address the importance of VM technology in DCN, then, highlights how the routing technology affects the performance metrics of a DCN, we then focus on comparing importance of switch-centric topology (a DCN topology where the routing technology is centered around the switches), lastly we address the routing techniques which are vital for performance.

2.2.1 Virtualization in tiered DCN

To deploy an energy efficient system in a DCNs, many methods have been suggested. The hybrid network topology i.e., utilizing servers and switches for packet forwarding was suggested. This system provided better results, by having a better computing performance and energy efficiency. Although using servers this way inhibits setting underutilized servers to sleep, because most switches have VMs running on them, and these VMs are mostly servers. Thus, the server load consolidation scheme has been proposed which limits the server's load to a subset, with a possibility of 83% energy savings in a DCN [27] [28] [85]. Because of the consolidation of the hybrid network, i.e., the VMs and switches for the server's workload, the energy efficiency of a data center becomes highly dependent on the topology used. As a result, more focus on the merging of virtualization for the server's in the architecture has been a recent trend in data center networks [4] [83] [84] [85].

Statistics show that server-only utilization for an entire data center consumes up to 30% of the energy used [85], this is because most servers are on and running for just a single

application, making switches to be at the on state and in effect consume energy. Hence, to maximize servers' utilization, by running multiple applications, and adopting virtualization technology using a VM for servers, whilst providing QoS, on the same machine has become adopted [38] [30]. The benefits of running servers with VMs are numerous: it's highly cost-efficient with benefits ranging from 30-40% over a DCN without virtualization [5] [40] [85], also saving energy of up to 60% compared to a DCN without virtualization.

This virtualization technology makes load balancing easier, combined with fault tolerance and continuity of service in case of failure of any server. Virtualization also adds better scalability to tiered data center, and reduces the administrative load, with ease of migration of service using VMs, from one switch to another [5] [31] [38] [39] [40] [85], although, the increasing server utilization using virtualization has its drawbacks, which is an increase in the heat gain density [40]. Nonetheless, this setback is insignificant to the energy efficiency and performance benefits derived from running a DCN with virtualization. As shown below, Figure 2.1 shows a leaf-spine DCN with an integration of VMs in servers (at the bottom) in the DCN. The benefits of this technology as explained earlier are quite numerous [42], which includes energy efficiency.

Figure 2.1 Cisco massively scalable data center with the spine leaf topology [4] [42].

2.2.1.1 Energy efficiency in DCNs using virtualization

Edward et al, in [39], emphasized the benefits of running virtualization technology in a data center. Their experiments outlining the ease or service continuity from one virtual server to another in a datacenter, for situations where virtual servers fail, thus, giving an energy efficiency of up to 60% over a DCN without virtualization. Antika et al, in [41] wrote a detailed survey of various heuristic algorithms, which can be used in deploying VM placements in cloud computing of a data center, with a focus on the benefits energy efficiency brings. Zang et al, in [54], has one of the earliest works of using TE for energy efficiency in a DCN. He proposed the use of MPLS-TE in the core of the DCN to manipulate the networks paths to have an efficient performance, such that idle routers and

links can be put to sleep. The approach resulted in an approximation of 50% energy savings, by combining OSPF and MPLS for TE in DCN.

2.2.1.2 Energy efficiency in DCN communication system

The survey on green energy in network systems by Bolla et al. in [53], showed various ways green energy are deployed in a network. These ways can be thus summarized:

- Using smart sleeping of devices via Ethernet links with a 37.5% reduction in energy consumption by IEEE 802.3az standards [20].
- Dynamic adaptation using idle logic resulted in a theoretical 80-90% energy savings.
- Dynamic voltage, which resulted in a power consumption reduction of more than 50%, although other factors were not taken into consideration such as latency and packet loss, etc. [53].

In [32], an elastic tree was used for shutting down underutilized networks. In another work, energy reduction by managing port rate and usage [33] [34] on Flattened butterfly topology was been proposed. TE has also been proposed for energy reduction [11] [12] [50] on FAT tree DCN. Another work on energy efficient data center Networks using SDN, thus having a Software Defined – Wide Area Network (SD-WAN) was proposed by [45] [104]. Kakadia et al. [35] [36] suggest selecting the number of servers and switches to be turned on and off by some algorithm in a FAT tree architecture using Openflow on SDN to increase resources accordingly, thus saving energy in a DCN.

2.2.2 DCN communication infrastructures

The infrastructure of a DCN plays a vital role in the scalability of the DCN. In general, the requirements for using a DCN topology is for the DCN to deliver peak performance to users, the following factors play a vital role in achieving the requirements, including fault tolerance, scalability, reliability, routing efficiency, low latency, high network capacity, and support of virtualization [84]. Currently, there are various classifications of DCN topologies that have been used, e.g., [3] [4] [5] [11] [24] [25] [56] [83] [84] [85]:

- The unconventional tiered (recursive-defined) topologies, such as B-Cube, Dcell, butterfly (3T), FiConn, FlatNet, SprintNet);
- The Conventional tiered Clos/Tree based (e.g., FAT tree, and leaf-spine VL2, Portland, F² tree) built from commodities resources and off-the-shelf resources;
- Hybrid network (e.g., c-Through, Helios);
- Direct network (e.g., CamCube, Small-World);
- Switch centric topologies; and
- Server Centric topologies.

But, for the purpose of this thesis, we would classify them into two: the switch-centric (Clos topology, Fat-tree, leaf-spine etc.) and the server-centric (BCube, MDCube, Scafida). In the past, the tiered topologies (3 tiered) were mostly used in DCNs with high-end switches, high costs, enhanced functionality, and higher energy consumption. On the other hand, unconventional topologies such as B-Cube [24] and D-Cell [26], butterfly all make use of low cost resources, and consume lower energy. But the Clos topology (switch-centric) consumes higher energy. Hence, there arises a trade-off, a better performing network will require tiered network, but a higher energy consumption, and an energy

efficient topology would shift towards the unconventional architectures. Nonetheless, to meet the current demand of a High-Performance Computing (HPC), in a datacenter, combining the virtualization technology that provides an improved performance as to latency, scalability, throughput, congestion reduction with the switch-centric architectures is favored because of its performance [3] [83] [84]. But, this type of topology has inherent issues which can be factored into:

- Oversubscribed links in high-end switches
- Problems with scalability (increasing or decreasing the number of servers).
- The routing design creates fragmentation of server resources.
- Spanning tree problem.

Most of these problems are addressed in the various improved Fat-tree technologies, proposed in [3] [83] [85] [88] [89].

2.2.2.1 DCN switch-centric topologies

With the high network speed and low latency required from a data center, HPC in data centers is on the fly. The switch-centric topology is one in which the routing intelligence of the DCN, is centered around switches, where each server in most situations uses one port of the Network Interface Card (NIC) to connect to the network. On the other hand, in server-centric DCNs, the routing intelligence is centered on servers, where switches are used as cross-bars and multiple NIC ports are used [103].

Of the Clos family DCN, the switch-centric topology is widely used by many data centers of recent, e.g., Google (Google Fat-tree), Facebook (leaf-spine, two-tier), Microsoft

[83] [87], etc. The reason is its improved performance in integrating virtualization technology for servers into the DCN, compared to the server-centric topologies. In the switch-centric family, the Fat-tree was initially being deployed. But of recent, due to its inherent disadvantages, this topology has been improved upon to the leaf-spine topology, google Fat-tree, diamond, F^2 tree, etc. The following sections highlight the main points in using these recently improved topologies.

2.2.2.1.2 Fat-Tree topology

This topology as shown in Figure 2.2, was originally designed by Al-Fares et al. [25]. in 2008, Fat-tree was designed as an extension or an improvement of traditional tree-tier DCN. It is one of the most popular clos network topologies designed some 50 years ago, for telephone networks. The hosts are connected to the access layer, while the aggregation layer is between the access and the core layer. The aggregation layer is designed in such a way to provide non-blocking multiple links of communications between any two hosts, in the network. Furthermore, in the Fat-tree topology, each n -port switch is connected to $n/2$ servers, while the remaining $n/2$ ports are connected to $n/2$ switches in the aggregation layer. Furthermore, there is no need of any special switches at the aggregation or core layers. This is why, one can use any cheap commodity switch to build a Fat-tree. This is as a result of the bisection of bandwidth at the aggregation and core layer forming a bipartite graph on both layers. Each flowlet has a unique path of all possible paths including redundant paths, to reach a common ancestor thus avoiding reordering of packets. A flowlet is a collection of TCP segments sent in quick succession. After reaching the common ancestor, it travels downward taking the only possible path.

Figure 2.2 A simple Google Fat-tree [adapted from 3, 25, 88].

Fat-tree has the following as advantages [3] [25] [85]:

- Fat-tree has identical bandwidth at bisections.
- They are known for their non-blocking nature, providing multiple redundant paths between any two hosts.
- High performance switches are not necessary in the aggregate or core levels.
- Each port supports the same speed as end host.
- All devices can transmit at line speed if the packets are distributed uniformly.
- It is highly scalable, k -port switch supports $k^3/4$ servers, e.g., using 48-port switches can accommodate 27,648 servers. Hence it is designed for large scale DCNs.
- Hosts request may not necessarily leave the POD (A pod is a constrained location of devices in a DCN, usually consisting of Access switches, Servers (hosts), and in some cases Aggregation switch), based on the look up table.

2.2.2.1.3 Diamond topology:

The diamond topology [92], as shown in Figure 2.3, was designed by Yantao et al. in 2013 as an improvement of Fat-tree, to replace all the aggregation switches of Fat-tree with edge/access switches, and connecting this access switches directly to the core switches. By doing this, the diamond topology has a 10% delay lesser than fat tree, for an average end to end route. It has the following advantages [92]:

- It has a better route lengths as well as lower end-to-end delay compared to Fat-tree.
- Its design reduces the average hops to 4.5 compared to 5 hops in Fat-tree for inter-pod routing.
- It is built to for large scale DCNs similar to Fat-tree, e.g., a 48-port switches can accommodate 27,648 servers.
- Traffic distribution is more even than Fat-tree, hence it has lesser congestion.
- The network is built to utilize its multiple path based on the FAR (Fault avoidance routing) lookup table, hence it is more robust and has network reliability compared to Fat-tree.
- Two types of switches used, the core switches and the edge switches.

Figure 2.3 A diamond DCN topology improved from Fat-tree [adapted from 83, 88].

2.2.2.1.4 Leaf-Spine topology

The leaf-spine topology, as shown in Figure 2.4, is a proprietary design of Cisco. The leaf-spine has been improved on by both Facebook and Cisco in recent years. Facebook introduced a ring for failure recovery on the spine layer as well as the access layer of the topology, and Cisco introduced higher efficient VX-LAN technology to improve its virtualization technology [91] [94] [96]. But in general, the basic infrastructure still remains the same, such that every leaf switch is connected to all spine switches in a full mesh topology [4] [91] [94] [95]. The spine layer forms the backbone of the network. It was designed for industrial purpose to address the issues facing Fat-tree and provide a simple, flexible and stable topology with fast convergence for the industry. By merging the aggregation and core-layer of the Fat-tree into a single spine/core layer as shown in Figure 2.4. Although, over the years this design has been improved upon, but the original design architecture still remains the same and is still widely accepted [91] [93] [94].

Figure 2.4 A simple Facebook leaf-spine topology [adapted from 1 and 87].

It has the following advantages [3] [83] [91] [93] [94]:

- It has a very high throughput
- It is easily scalable
- It has a high link utilization
- It is not easily affected by oversubscription ratio

Although there are other developed versions of this topology, for accommodating more servers and VMs. This thesis focuses on the plain leaf-spine topology [91] [94] [96]. Since it's our topology of choice, more of its benefits would be highlighted in section 2.3 below.

2.2.2.1.5 F²Tree topology

In 2015, Guo et al. proposed the F²tree [95], as shown in the Figure 2.5. This topology was developed principally for a rapid failure recovery in a DCN, as an improvement of

existing topologies in network failures. F²tree ring across switches has been adopted in some topologies e.g. for leaf-spine by Facebook, VL2 [95] [96], etc. The main idea of this topology is to introduce a ring across the core and sets of aggregation switches, which create immediate access to backup routes during network failures. As a result, the time for link recovery from network failure is reduced, making the network fault-tolerance. However, it has fewer hosts compared to the diamond topology.

Figure 2.5 A F²Tree DCN Topology [adapted from 83 and 95].

2.3 Comparing the leaf-spine over some switch-centric topology in a data center

There are many factors that are taken into consideration in developing or choosing a DCN topology. A summary of these factors is listed below, this summary is derived from points listed in sections 1.2, 2.2.1, as well as the focus of these thesis in energy savings and networking efficiency. The following factors are paramount in designing a DCN topology:

- The number of switches or servers required to be active.

- The number of links required.
- The link utilization ratio.
- The throughput of the network.
- The cost and energy expended on the network.
- The scalability and performance with virtualization technology.

In a DCN, it is essential to ensure that all devices can transmit at the line speed of the network or the link speed of network fragment layer which are the access layer (between the leafs and the hosts) and the core layer (between the core switch and the leaf switch) thus offering a subscription ratio of 1:1. It should support $\{\frac{k^2}{4}\}$ servers, which results in high scalability [4] [56] (where k is the number of ports on the switches). Tables 2.1 – 2.3 show essential metrics adapted from both [3] & [83] from the experiments in comparing these DCN topologies.

Table 2.1 A comparison of 3 DCN (adapted from topology design [83])

	<i>Fat-Tree</i>	<i>F²Tree</i>	<i>Diamond</i>
<i>Core switches</i>	$k^2/4$	$\frac{k^2}{4} - k/2$	$k^2/4$
<i>Aggregation Switches</i>	$\frac{k^2}{2}$	$\frac{k^2}{2} - k$	$k^2/2$
<i>Edge switches</i>	$k^2/2$	$\frac{k^2}{2} - 2k + 2$	$k^2/2$
<i>Hosts</i>	$\frac{k^3}{4}$	$\frac{k^3}{4} - k^2 + k$	$k^3/4$
<i>Links</i>	$3k^3/4$	$7\frac{k^3}{8} - 13k^2/4 + 3k$	$3k^3/4$

Table 2.2 A comparison of cost and energy efficiency (adapted from [83])

<i>Topology</i>	<i>No. of switches</i>	<i>No. of links</i>	<i>No. of hosts</i>
<i>Leaf-spine</i>	30	260	156
<i>F²Tree</i>	92	580	160
<i>Diamond</i>	80	384	128

Table 2.3 Performance comparison with increase in the number of servers [3] [83]

<i>Performance metrics</i>	<i>Leaf-spine</i>	<i>F²Tree</i>	<i>Google Fat-tree</i>	<i>Diamond</i>
<i>Latency</i>	Lowest	High	Average	Average
<i>Number of Switches/links</i>	Low	High	Average	Average
<i>Link utilization ratio</i>	Highest	Lowest	Average	Average
<i>Cost/Energy required</i>	Lowest	Highest	Average	Average

The tables above summarize the various characteristics of recently deployed switch-centric DCN topologies. Looking at the equation 2.3 below as well as tables 2.1-2.3 we come to a simple conclusion that the leaf-spine topology is the preferred topology when considering cost, energy efficiency, link utilization as well as performance, in relation to other performance metrics. In addressing the inherent issues facing Fat-tree topologies, the recent topologies developed, such as diamond, F²Tree, has resulted in higher number of switches required to run the DCN compared to the Facebook leaf-spine DCN, for an approximate number of hosts and servers as shown in Table 2.1 and Table 2.2. Table 2.3 shows the performance of recently used topologies in DCN. Yedder et al. [25] [83] summarized the efficiency of leaf-spine topology in a DCN over other topologies in the clos family for deploying virtualization technology. In summary, leaf-spine is cost effective, energy efficient, and performance efficient to link utilization ratio, although not

as robust as to fault tolerance compared to F²tree. One major modification of the leaf-spine DCN compared to the Fat-tree and other topologies (3-tier) is that the aggregation and core-layer are merged.

The cost per usable port in a DCN is calculated as follows:

$$\frac{(Cost\ of\ switches + Power + Optics + fibre)}{(Total\ no\ of\ nodes\ X\ oversubscription)} \quad (2.3)$$

The following points are some advantages of using the leaf-spine topology, over some other topologies:

- Unlike in 3-tier network, in 2-tier clos layered network (leaf-spine), the delay and latency on the network are easily predictable, because from east to west the hosts are all equidistant.
- Due to its infrastructure, there is always an easy immediate alternative path for the packet traffic. Hence network congestion can be reduced. It has very high link utilization, thus enabling it to scale.
- It is also designed for auto-scaling. The network can easily scale up by adding a spine switch with a couple of leaf switch to the network, to meet the spike in network demand, and as well scale down when the network is not in high demand.
- For the same hosts and leaf switches, it can use, 10Gbps between the access links and leaf switches, as compared to fat tree which uses 40Gbps because it's a 3-tiered infrastructure. Cisco in [4], shows that the current power consumption of 40Gbps is 10X more than a 10Gbps link [4] [42].

The general rule of thumb as shown in equation 2.1, equation 2.2 and equation 2.3, is that in the higher the DCN tier the higher the energy expended, as well as the cost. Furthermore, it is essential to accommodate the oversubscription ratio of the network bandwidth, such that each device should be able to communicate with other devices on full network bandwidth.

As shown in [3] [56] [83], one disadvantage in using the leaf-spine network is also in its advantage. Due to its high scalability, it can scale up to the extent that the number of leafs used in the network could go beyond the number of spine ports available to accommodate them. The number of leafs used in this topology can only increase to the extent of the number of ports available on each spine switch. This brings some limitations as to scalability.

Hence, we can see that the leaf-spine topology as compared to other topologies is more energy efficient as well as cost efficient. This is because, the equation 2.3 and table 2.2 shows that, if the same set of switches are used, for the access, aggregation and core, for all switch-centric topologies used in DCN highlighted in section 2.2.2.1, the equation shows the cost to run a leaf-spine is lesser to other aforementioned topologies.

Looking at all the points highlighted thus, far, we can conclude that, the leaf-spine topology is still a preferred topology to be used in a DCN. Some improvements may be added to the design, for quick backup for fault tolerance, as well as better interconnectivity between multiple leaf-spine architecture as used in [91] [94] [96]. Because the basic

structure of this topology is still been widely used, this thesis adopted the use of these topology in developing the SR-GSSMS.

2.4 Software Defined Networking (SDN)

In recent years, the interest in SDN has risen as a result of the diversity of various ways SDN can be applied to manage, control, integrate and deploy a network. SDN has become an emerging technology short after due to its software architecture and advantages. This has resulted in numerous ideas introduced into traffic routing. This technology has also been embraced by some data centers, resulting in SD-WAN. These brought about the merging of SDN technology to the traditional IP technology, as embraced by both Google and Cisco [73]. It could be merged to an existing traditional network, such as SD-WAN due to its dynamic nature. Furthermore, another advantage of SDN is its ability to simulate a real physical network, which creates the basis for running numerous experiments. All these advantages of SDN is because of its software infrastructure [51] [104].

Figure 2.6 SDN architecture with APIs communications [87] [51] [104]

As shown in the Figure 2.6, the SDN service layer provides functions including routing, firewall, equal cost multi-path, MPLS etc. While the switches southbound connect northbound to the controller through a northbound interface called OpenFlow. A simple picture of this infrastructure is by its nature, similar to a master-slave API infrastructure. Thus, in general, especially from Openflow 1.1 and onward, we have in the Northbound, the application layer, where so many applications and network functions for programming, deploying business logic, in the network resides. These communicate to the controller in the central, via northbound APIs. Next is the control layer, here is where the centralized control for SDN switches and forwarding rules generates. In general, this layer is termed the controller, the central brain. It communicates rules, and roles to the switches via the southbound API, and connects to the northbound application functions via northbound-

API. The southbound layer is for the data plane and OVS switches for pushing flows to hosts.

It is designed in such a way that there is a separation in the forwarding plane and control plane of the network. This provides a dynamic control of network functionalities, such as high-bandwidth, effective traffic engineering, MPLS, Segment routing, application-aware routing, etc. These network functions are all located in the northbound of the SDN infrastructure. This eases the deployment of programmable network functions in a SDN system. The control plane is handled by a centralized system which is the controller, whilst the data plane and forwarding plane are south bound, where the switches and host reside.

The advantages of SDN over traditional routers and switches can be summarized as follows [87]:

- The network engineer is able to configure the network design using software programs instead of thousands of lines for router and switches configuration manually.
- It is possible to replace expensive switches and high throughput switches with large numbers of commoditized switches. This is because SDN enables higher throughput more flexible flows, with less payload, less operational and capital expenditure.
- The controller has a global view of the network, this makes it possible to easily update deployments of network functions, as well as deploy these functions over the network.

There have been so many controllers that have been developed over the years. But they all have in general, a similar infrastructure. Our focus is on the SDN Open Daylight (ODL) controller. The numerous APIs of ODL makes it much easier to deploy many networking functionalities. We would not describe in details these controllers here. But Figure 2.7 presents a general idea of the network functions and structure of the ODL controller. Please refer to these links for further explanation in understanding the differences between traditional network and SDN network [59], also for further details understanding the infrastructure of ODL refer to this links [61].

Figure 2.7 shows a variety of APIs, which could be installed in ODL. The APIs in ODL could be proactively deployed, actively deployed as well as reactively deployed, in a network, depending on the use cases for the network engineer. This is where the needed deployment of algorithms can be found and adjusted. The YANG, LACP, OVSDB, PCEP, RESTCONF etc. APIs were mainly utilized in this thesis. For further details, please refer to the documentation, for both developer and user in ODL website [61].

Figure 2.7 The SDN ODL infrastructure [YANG, LACP, OVSDb, PCEP, RESTCONF, API'S] [60]

Hence, after the introduction of the environment above, we would be explaining some of the features used in this thesis, such as MPLS-TE, Segment routing, spine reduction algorithm, as well as auto-bandwidth configurations and rest APIs. For the rest of this chapter, we would explain Traffic engineering (TE), MPLS-TE, and segment routing. The next chapter would cover details of the algorithms deployed for the SR-GSSMS.

2.5 Introduction to Traffic Engineering (TE)

Over the years, a plain vanilla network is incapable of handling congestion is traffic, deploying, Equal Cost Multi-Path (ECMP) on networks, Quality of Service (QoS), fast network recoveries from failures. Hence, various methods have been developed to improve network performance, and mitigate the issues listed above, the term used for these methods

developed is called Traffic Engineering (TE). In simple terms, TE is the ability to steer a traffic from its default routes to a new route with the goal of improving the network performance and functionality. Of the methods that have been used to achieve TE over the years, our focus is from the Multi-Protocol Label Switching (MPLS), and onwards.

2.5.1 MPLS Traffic engineering

Traffic engineering as explained earlier is simply deploying various methods to improve the performance of a telecommunication network traffic, such that the network becomes predictable, resilience to network failures, node failures, and dynamic by nature with a better performance. These characteristics are very vital for a service provider in a data center and an Internet service provider (ISP) deploying a WAN network. MPLS provides an integrated platform for combining many characteristics of TE in a Network [62], which is modeled after the Resource reSerVation Protocol-Traffic Engineering (RSVP-TE) [74].

In a simple explanation, routers and switches all have labels, which can be seen in the MPLS data-plane payload. In MPLS technology, these labels are included in the advertisements sent to the edge routers in the network, called the head Label-Switched-Paths (LSPs). The head LSPs uses the configuration information derived from the advertisement of a packet to determine the paths to route the flow. Thereafter, the paths are forwarded downstream of each LSPs depending on the Forwarding Equivalence Class (FEC), used for the flow. The packet while going downstream, carries out the MPLS actions of popping, swapping and pushing of labels attached to it, depending on the position

of the routers/switches in the LSPs. Routers at the beginning, does the pushing, at the middle swapping and popping, while at the end popping all these occurs during the flow in a LSP. MPLS-TE brought the following advantage to traffic engineering:

- It dynamically maps a network traffic flow using switches labels to predefine a path
- Depending on the constraints configured in a router, such as priority, shortest path, bandwidth (BW), QoS, MPLS-TE can be used in automatically forwarding packets across a network.
- It can recover from link and node failures, using pre-determined constraint to reconstruct new routes using the Label Switch Paths (LSPs) [63] [64] [74].

MPLS can be said to be a layer 2.5 protocol, in that it is a combination of both layer 3 and layer 2 protocol technologies. A plain IP network is incapable of efficiently utilizing its resources, nor efficiently directing packets based on the network properties available. Hence the introduction of MPLS. To deploy MPLS in a network, based on the Interior Gateway Protocol, (IGP), when a flood of packet reaches the head of a network path in some cases the head of LSP (looking MPLS as layer 2 protocol), or as a layer 3 protocol the head of a virtual link, these nodes, uses the configured constraints, to calculate and determine the various LSPs to route each packet.

In calculating the shortest path function (SPF) for the LSPs, the link state IGP uses (OSPF or IS-IS), this enables the dynamic nature of MPLS. The TE carried out on the network is mainly done at the ingress of the network. This is where each path switch label

for the various flows are determined and used for forwarding the flow. After packets are classified into FECs, the Label switching forwarding mechanism in the data plane (popping, swapping, and pushing-on of labeled paths) makes use of a layer 2 characteristics. Such that the multiple paths established by the Resource Reservation Protocol (RSVP) signals are used to forward packet into multiple LSPs [64]. What makes MPLS technology much probable is its ability to create tunnels in networks, this functionality is very fundamental to the success of MPLS, as a result, in complex networks and a network requiring ECMP, many issues surrounding MPLS arises. This will be highlighted in section 2.6 below.

2.5.2 MPLS-TE in SDN networks

A traditional MPLS network has combined the control-plane and data-plane on the same layer. Giving much functionality and scope for the control of flows to the edge switches and routers for traffic engineering. Nonetheless, we see the same protocols, and functionality shared amongst most networking devices. The information carried is vital to some routers (at the edge) but to many routers, the control plane payload is irrelevant, what is required is only the data plane for forwarding packets, unfortunately, all routers have to carry all the payload of information, be it vital or not. Hence, in the traditional network, every router is its own decision maker, with some, having a very high payload for just forwarding of packets, to a neighboring router. On the other hand, in Open Flow, there is the abstraction in simple words, a separation, of the control plane and data plane, such that the controls, rules, routing and decision making are all placed in the control layer of the controller. The controller mediates between the northbound applications using northbound

APIs and the southbound data plane using southbound APIs. While the data-plane simply handles the forwarding of packets [63]. Figure 2.8, shows the contrast of the traditional MPLS-TE network over the SDN. Showing a higher payload of many layers of protocols to be used by every switch in a traditional network. On the contrary, all these protocols, are handled in the controller of the network, of SDN. Thus, giving SDN controller more advantages in deploying TE, with a lesser payload over the traditional network.

(a)

Figure 2.8 Traditional Network IP/MPLS vs SDN MPLS (a) Abstraction of control-plane in SDN (b) comparing SDN MPLS with map abstraction [63].

With the MPLS plane abstracted to the controller in SDN, this makes deploying MPLS functions much easier as shown in Figure 2.8 (a) and (b), because the controller has a global view of the entire network. The time delay in establishing in each router an entire view of the network for decision making is out. The SDN controller has a global statistic of available resources, hence constraints needed to be deployed for MPLS-TE on the SDN network, could be dynamically used. Open Flow 1.3 onward supports this technology in Open v Switch OVS) [65]. When OVS is used as an Open flow switch, the TE functions such as popping, pushing, swapping of labels can be easily done, as well as tunneling between the networked switches.

2.5.3 Benefits of SDN-TE over MPLS-TE

To appreciate the concepts of SDN Traffic Engineering better, looking at Figure 2.8 shown above gives a better insight as to difference between traditional MPLS-TE and

SDN-TE. In a traditional MPLS-TE, the same link state routing protocols are used by each router only with an extra payload of information for the MPLS_TE routing. Each router uses this information to calculate the best routes to forward a packet. In the case of calculating the shortest path for a network, the network engineer could vary the link weight of a node(s) or destination hence enabling Traffic engineering to be deployed by the edge router or the head LSP. Each router's view of the network in this environment varies, because it is based on the advertisements the router receives. Each router has its own network map used in forwarding packets. The constraints needed for Traffic engineering, cannot be deployed on the fly, but in some situation, needs to be configured, dynamically or statically.

On the other hand, in SDN, taking advantage of the Map-abstraction of network functions into the northbound layer of the control layer, and a global network view of the network. The controller can make use of applications such as MPLS-TE, bandwidth, Quality of Service (QoS), shortest path forwarding (SPF), access control etc. dynamically and actively, by using APIs to the southbound data plane for forwarding network traffic. Thus, TE engineering for networks becomes much easier, because these functions could be programmed, into a network. This reduces the payload on each switch for carrying all the protocols required for forwarding packets and deploying TE. With the dynamic nature of the SDN controller, the rules for forwarding packets by each switch are sent by the controller. Hence switches have a lesser payload of information, and processing traffic engineering takes less CPU load, compared to a traditional network [63].

With these advantages in mind, deploying TE, becomes lightweight in a SDN environment. More details on SDN TE and MPLS-TE, the pop, swap actions and more can be found in the following references [63] [66] [67]. Focus, on the remaining part of this chapter, would be on explaining the importance of the network functions used in this thesis work, such as bandwidth on demand TE, QoS TE in SDN, and segment routing.

2.5.4 Traffic engineering using auto-bandwidth

To implement auto-bandwidth using TE on traditional IP network, the edge routers are usually configured to get bandwidth usage information from every LSPs, and tunnels on a timely basis, the timing vary depending on the network engineer's configuration. Based on this information, the router makes bandwidth reservation on the LSPs. When there is a change of BW reservation on a path, this information is relayed first across the LSPs upstream and thereafter over the network.

On the other hand, deploying auto-bandwidth in an SDN environment, with the abstraction of the control layer separated from the data plane. The switches, been controlled by the southbound API of the controller, can easily make use of all the traffic engineering functions, such as MPLS-TE, auto-bandwidth management, QoS, SPF, admission control, link-state, much easier and with lesser payload as compared to a traditional network. Hence, the controller can be programmed to make the necessary constraints decisions for forwarding in the data plane. Since the data plane is unaware of the BW reservation for a flow, the controller does all the necessary calculations and adjustments for TE forwarding

table for each switch, which is relayed southbound to the data plane for all LSPs. This can also be deployed on a TE tunnels as well, in SDN.

2.5.5 Traffic engineering using QoS

The principle for deploying end-to-end QoS using MPLS-TE on a WAN and DCN are similar, in that on all levels of QoS consideration, the per class base consideration of QoS is the same, depending on the mechanism used. There are many QoS mechanisms available, such as Class of service QoS, which is used in separating packets into different classes. This comes in very helpful in cases of congestion and transmitting multiple packets class over a network, and ToS in QoS for differentiated service code point (DSCP), also known as DiffServ. This comes in very useful, because it is a combination of packet marking and well defined per-hop behavior (PHB). It provides a packet with appropriate jitter bound, delay bound and bandwidth bound control providing a scalable QoS for packets.

Differentiated Services architecture (DiffServ) separates packets by packet marking, then prioritizing packets flow in switches on a per-hop behavior (PHB) based on the classes. Congestion could result if this is the only protocol used, for TE. Because there is no mechanism to propagate this information upstream in the network. To avoid this, the TE Interior Gateway Protocol (TE-IGP) has the payload of informing the edge routers of these changes. TE controls packet hop-to-hop basis. Hence, a reserved tunnel, with available BW, could be used to change the path thus avoiding congestion. To properly deploy QoS in a Network, both Differentiated services and MPLS-TE are needed. This process is called DiffServ-TE. If QoS by DiffServ is to be deployed on the network, a

policy based router (PBR) would be needed on a traditional network, yet still, all these configurations are configured manually [75] [76].

On the other hand, in an SDN Network, all these separate functionalities are in the control plane of the controller, hence the deployment of QoS-TE is much easier and can be automated, and dynamically deployed to control the data plane forwarding mechanism of the network from the controller.

2.5.6 Traffic engineering using ECMP and UNECMP

For the purpose of this write-up, Unequal cost multiple path will be termed UNECMP. UNECMP, this is the term used in routing flows across various network links, especially links with varying costs. In so doing, the load balancing across the links is said to be an unequal-cost load balancing (UCLB). Hence the weight of the flow is said to be balanced across the links unequally. Cisco, have written numerous information on how to implement this TE, over their routers, using MPLS-UCLB, for more information please refer to cisco documentation in [103]. ECMP, on the other hand is a TE technology, such that, the load of a flow is shared equally across links which have the same cost. By so doing, the load balancing in this case, is said to be equal cost load-balancing (ECLB) [90].

Advance traffic engineering with MPLS, or Segment routing, can be done on links, such that, links that have unequal cost may be forced to have equal cost, in other to load balance the flow. In the case of this thesis, UNECMP is traffic engineered can be deployed

over links with equal cost and vice versa. By so doing the network engineer has better control over the flow of the network.

2.6 Introduction to segment routing:

Despite the introduction of MPLS, there have been many issues in MPLS networks, because of the fundamental nature of MPLS engineering, which majorly uses tunneling for traffic steering. The following are issues which are persistently difficult to mitigate in MPLS-TE [74]:

- The poor balancing of RSVP-TE to provide an ECMP network is one disadvantage. To mitigate this fundamental incoherence, thousands of RSVP-TE MPLS tunnels are needed, which makes a simple network, very complex, making traffic monitoring difficult. In Segment routing (SR), a single tunnel can be used to capture multiple paths, making it into one segment used in the network. The simplification of SR makes TE much easier.
- Due to the heavy payload in routing, of which in some cases the information required is only in the data plane for forwarding packets, these issues of scalability of the control and data plane is on the rise, in traditional MPLS-TE.
- With distributed computation on RSVP-TE, traffic placement in the network is very unpredictable. Furthermore, in MPLS-TE we see a poor utilization of network resources and a higher lag in network recovery and link aggregation.
- With SR-TE, there is no need to maintain per-application and per-flow state, the instruction embedded in the segment routing packet is used instead.

- With segment routing-TE, one can also simply integrate auto-bandwidth in the design, ECMP or Unequal ECMP (UNECMP), OAM support during configuration such that the source can deploy these functions dynamically over the network.
- When combined with SDN, the network engineer can program in the controller how and when the SID are to be dynamically selected, especially when the network bandwidth is in demand.

With the issues facing MPLS-TE and the rise of SDN using the concept of a global view of the network. SPRING, source packet routing technology, known as segment routing was developed. This new routing technology helps mitigate most of the problems facing MPLS-TE [74], as well as provide an easy platform of integrating traditional networks with SDN networks.

It should be noted that in this thesis, segment routing is sometimes used interchangeably with SR.

2.6.1 Segment routing

In simple terms, segment routing is just Source Packet Routing in Networking (SPRING) technology. In that, the source of the packet (a host or an edge router) uses the available information of network parameters, to predefine the segments (labels) to be used for a particular flow. By its nature, the shortest path is engraved in SR nodes including ECMP which is in support of IP design.

A router using an enabled SR network can either use IGP or a segment identifier (SID) to determine the shortest path to its destination. The downstream nodes or routers simply push the packet based on the Segment Identifier (SID) embedded into the flow, using either push, swap, or pop action depending on the node in the SID [68] [69] [74]. It should be noted that a router can play a host to multiple SIDs, giving segment routing the needed flexibility and advantage in deployment.

Thus, after dividing the network into different segments, such that each path or link is assigned a SID, the source router for a packet uses the information in the control plane to control the flow of a packet to its destination, directing the flow over various segments of the network to its destination [77-79] [82]. The SR control plane could fit into either the IPv6 header, creating an extension of the IPv6 data-plane or it could seamlessly be applied to the MPLS data-plane. As explained earlier in section 2.5, the MPLS data-plane, when used for SR control plane, is the same. Only names and functions are changed. The labels are called segments, and label stacks are called segment list [74]. Segment routing in IP/MPLS architecture makes use of the same source routing IGP protocol (IS-IS/OSPF) with an extension. The labels of the set of nodes or routers used to form a path in the network are called segments, hence the name segment routing [77].

For a balance of efficient centralization and distributed intelligence, on an end-to-end basis, segment routing gives a complete control of network forwarding in using the available instructions in the protocol payload to route a packet. Thus, the same push, swap, and pop actions of MPLS labels are carried out on SIDs in routing a packet. The SID can

be either globally unique SIDs or locally significant SIDs. It is recommended to make use of globally unique SIDs for large networks. If a segment is not in the segment routing global block, it is considered to be of local significance with a local SID.

Furthermore, segment routing is simple, scalable, and tolerant to failures in a network. When a link failure occurs, the packets are simply rerouted from the node causing the failure, the fast reroute makes an intermediate solution (it becomes an intermediate SPRING node) recalculating the paths from itself to the next hop node, and administering this in the traversing path label for the traffic. A full explanation of these procedures as well as various applications is given in [77] [79]. There is no need for BGP route injection in segment routing, because, while using a globally unique SID for segments, the Path Computation Elements (PCE) router simply routes the packet to a destination by deploying a segment to be used. Hence, segments can be said to be sub-paths that a router combines to form routes to a destination. It utilizes bandwidths efficiently and provides path protection by default [74] [77].

2.6.2 Types of segments

There are two main types of SID, the prefix, and the adjacency SIDs. The following explanation are derived from the following references [72] [69] [74] [79] [82].

- *Adjacency SID*: Just as its name adjacent meaning nearby, adjacency SID, is said to be of local significance, i.e., locally unique. It is usually the SID between two routers, but specific to one router, hence it's unidirectional, with one hop. In a complex network, using only adjacency SID, the segments selected do not results

in SPF, as shown in Figure 2.9. The routes used in this protocol do not have SPF, as shown in Figure 2.9 below.

- *Prefix SID*: This is a globally unique SID, it is calculated by a router or the SPRING source using the IGP protocol of the network. When used, it calculates the shortest path to the prefix, it has more than one hop. Figure 2.10 shows prefix SID that has been used to route packets with SPF in the network.
- *Anycast SID*: This is also a globally unique SID; it is an IGP shortest path (least cost) to a non-unique prefix.
- *A node SID*: This can be said to be a segment connected to a loopback that identifies a specific node. It is globally unique SID, and computes the shortest path by default in forming segments, used in routing.

In carrying out SR, the following actions similar to MPLS are used:

- **CONTINUE**; this is used to simply forward a flow to the next active segment in SR. The NEXT is used to mark the proximal active segments as the active segment.
- **PUSH**: this is used to add segments on the SR header in the control plane and chose the segment which would be active [74]. As deployed in many networks as well as used in this thesis work, a combination of both the node SID and adjacent SID is used for the segment routing with focus on bandwidth and link utilization.

The IGP advertisement is vital, it is shared across the network by default, such that any router or node requiring to form a segment would use this information to create its segment path from its control plane, In the case of adjacent SID, each router sends across information of their adjacent node, while in nodal SID, each router sends across

information of their global view of the network. It is this information that is used to define routes (segments) deployed over the network in segment routing as shown in Figures 9 and 10 below.

Figure 2.9 Adjacency SR operation locally unique segments (adapted from [69] [74] [77])

Figure 2.10 Prefix SR operation globally unique segments with shortest path (adapted from [69] [74] [77])

Figure 2.9 and Figure 2.10 above show adjacency and nodal SR, respectively, in action. Taking a close look at Figure 2.9, we see that despite there are so many shortest paths from R2 to R1 when using adjacency segments, this is not taking into consideration. Instead, only the adjacent segments are used to form path from the source to destination. Hence, this does not results in SPF in its computations. On the other hand, in Figure 2.10, nodal SID computes many segments from any node to any node in the network, the SPF, is included and in many instances, is of preference. The legend of red arrows shows the shortest path from any node to R1. While the black arrow legend shows the path from R2 to R1 via R6. The resulting path in Figure 2.10 uses less segments, and can be applicable to routers located to different continents. It should be noted that, routers advertising the same anycast node-SID, *must* have the same SRGB. Hence, we see change in the prefix SID of the same segment heading to R4, in Figure 2.10 above.

2.6.3 Segment Routing in a Leaf-spine DCN

In a traditional DCN, with no SR enabled, due to the network architecture of a leaf-spine DC, the default best routing protocol used is ECMP for routing. This makes load balancing easier in the network, because the next hop for forwarding a packet to a destination could make use of multiple best paths available. The downside of this routing technology is that load balancing elephant flows in a DC becomes cumbersome, resulting in congestion on the links, higher latency and bottle necks. Furthermore, when a failure occurs on a link, and a new route is selected, since ECMP is deployed, this adjustment results in a complex rerouting of flows in the network, to balance the traffic in the DCN as well as notifying the upstream nodes for route re-aggregation.

But, when segment routing is used in a DCN, the complex re-routing in case of a link failure does not occur, because by default fast reroute is a constituents of anycast SID and node SID. Hence, no further protocol is needed to adjust the settings on the network. When SR-TE is combined with SDN, the deployment is much easier, dynamic and can be preprogrammed into the controller for deploying SR-TE on the network, IN a traditional network, it could be configured in the router with lesser number of lines. In SDN, Ghuman et al. [50] showed in his dissertation, how segment routing TE can be used to reduce the energy consumption in the DCN, by reducing the number of active links on the network, and redirecting the traffic flow, in the network using classes and network functions that works with the controller on SDN network. Although, his work was based on the Clos Fat tree topology, which is lesser energy efficient than leaf-spine architecture, as explained in the section 2.3 above, the approach was able to achieve a good energy-efficient system on SDN as compared to when it was not. This idea and algorithm will be efficiently combined with the GSSMS algorithm for leaf-spine topology [1] to derive a much efficient routing paradigm called the SR-GSSMS.

Chapter 3: Segment routing-green spine switch management for DCNs

In this chapter, the focus will be on explaining algorithms used for the SR-GSSMS. To the best of my knowledge, this is the first-time segment routing algorithm is deployed on a leaf-spine DCN with SDN, with a focus on auto-bandwidth management, combined with energy efficiency. Chapter 2 has highlighted the importance of segment routing in traffic engineering, showing how it can be used to maneuver traffic for an improved performance in a network, as well as emphasizing the benefits of using SR-TE over MPLS-TE.

The SR-GSSMS algorithms will be discussed in this chapter, showing how the various algorithms fit into the proposed SR-GSSMS. It will further show how the SR-GSSMS can be deployed in a DCN, to derive a highly efficient system, in terms of performance as well as in energy savings. The DCN topology used for our analysis is the leaf-spine topology, which has been explained in Section 2.2.2.1 and Section 2.3.

3.1 The SR-GSSMS

The basic concept of the Segment Routing-Green Spine Switch Management System can be summarized, with the Figure 3.1 below. The system is divided into four major modules.

- The *controller module*, which is central and handles the core connectivity of the entire system.
- The *network monitor module* is used to monitor the traffic and switch utilization of all spine links and switch utilization thresholds.

- *The segment routing or routing module* handles the routing functionality of the network, in which either simple vanilla routing could be used or the segment routing which achieves much better results.
- *The power algorithm module* helps in increasing the energy efficiency of the system by taking down or bringing up links and switches, which creates the avenue for energy savings.

Figure 3.1 Green energy management system for leaf-spine architecture (adapted from [1])

3.1.1 Important highlights of the SR-GSSMS

A simple explanation of the entire system, when segment routing is deployed can be summarized as follows:

- The controller is activated listening for network connections. The *controller* is designed to listen to network information from the network monitor module with APIs, on a timely basis. The frequency of this is not predetermined; it is depending on the network engineer.
- Depending on the size of the network, especially the ratio of the spine to leaf switches, the higher the number of links and switches the longer it takes to connect the network to the controller. For example, for a small network of 40 switches, it takes less than 5 minutes.
- When a network is deployed, the *controller* first configures end-to-end QoS on all switches on the network. The approach makes rearranging flows in a switch easier, due to the making with DiffServ.
- Next, UNECMP is deployed in selecting the paths or segments on all switches.
- The *controller* chooses two links, as shown in Figure 3.2 below, for all flows in the forwarding table, which are dumped on the switches. This links are segments used in the network for flows.
 - One link is made active, while the other link is dormant and set to be the backup link for immediate fail over (the alternate link selected, must be one of the links of the active switches). The backup link is made available in cases of congestion, link failure, and load balancing.
- The controller is set to receive information from the network monitor every 2 minutes, this time is a variable which can be set by the network operator.

Furthermore, in any network, there are bound to be anomalies, which creates distortion in a network. A network is meant to be built to handle unpredictable situations, so as to be able to handle network anomalies properly. In a DCN this is very important, because it is designed to provide services to millions, the network has to be available always.

Figure 3.2 Network state highlighting backup links

When anomalies occur in a DCN, the effect has to be insignificant, such that the time for network failure is reduced to a minimum, and not necessarily noticed, by customers. The SR-GSSMS is designed to meet these requirements to properly handle poor network conditions, be it a forced network anomaly (such as auto-bandwidth and power module condition to reduce links, and switches) or an actual anomaly (such as congestions, link

failures). The following points highlight how the *controller* handles these anomalies on the leaf-spine DCN:

- Congestion on link: When congestion occurs on the link, the failover link is immediately utilized, and this failure is sent to the *controller*, which selects another backup link (segment) for the flow for immediate failover, and updates the forwarding table of all switches affected, in the background. This update is usually a broadcast update. Then the *controller* aggregates the links on the network, by aggregating the link using LACP, the network is set to be stabilized as if there was no congestion, in that, the new forwarding table updates is resonated across the network. The link selected is from one of the active spine switches. When the congestion is cleared in the network, the *controller* either load balances the flow over both links, or gradually transfers the flow from the back up link to the initially link selected.
- Link Failures: The approach in link failure is very similar to how the network handles congestion. Only, if the failed link is not recovered, the network would remain the same, with a new backup link chosen by the *controller*. During this process, the *controller* re-orders the links with the new forwarding table.
- Auto-bandwidth on network: In handling, excess bandwidth generated for a flow, the network approach is proactive, the *switch* selects the failover link and load balance the flow over the initial link and the secondary link (fail over link). This process triggers an API in the *controller*. The *controller* then creates new back up link. In other words, two active links and one backup link will be used in case the bandwidth been used is not sufficient enough. During this process, the *controller*

re-orders the links with the new forwarding table on the network. This process works in hand with the *power module* state of increasing or decreasing the average number of number of active spine switches (ANASS). When the number of links used reaches the pre-configured threshold, the *controller* next monitors the switch utilization threshold to determine the activation of a new spine switch in other to meet the excess demand on bandwidth. For example, in Figure 3.2 above, the number of inactive links is low. As the bandwidth continue to increase, the *controller* gets the switch 4 ready to be activated. The activation of switch 4 depends on the increase in the switch utilization threshold of switch 5. If the utilization of node 5 goes beyond the threshold for a set period of time, the node 4 becomes activated, then links in node four becomes backup links. In a similar way, when the bandwidth generated becomes low, the controller follows the reverse process. First deactivating links, updating the forwarding table with broadcast update, then deactivating switches, until the network is returned to its initial state. If the switch utilization gets lower the controller keeps reducing the ANASS until it gets to the minimum set at 2.

3.2 The SRGSSMS Modules

The points highlighted above can be thus described *the controller* is activated behaving like a server, listening for network connections, with the *network monitor* northbound APIs in place for deploying network functions, directed southbound to the data plane. When the network connects to the controller, it calculates the shortest path (depending on which *routing algorithm* if it is plain Dijkstra or *segment routing*. When the routing module is

SR, at the start, the network deploys unequal cost multiple paths (UNECMP) on all switches south bound; this ensures that more links can be deactivated or put dormant, although it must be highlighted that the controller is designed to dynamically select two paths for every flow to any destination for all leaf switches. One is used actively, and the other is placed as immediate failover link during any link failure, as well as could be triggered during congestion in the network, it is also used essentially in cases of load balancing, to handle excess bandwidth in a flow, this triggers the auto-bandwidth settings.

This approach makes it possible to isolate underutilized links, and switches in the network. When the bandwidth for a flow is higher than the threshold set for a link, SR will be triggered for auto-bandwidth management, the failover link is triggered to handle the excess demand, and if this link is not sufficient, the controller selects another active link with available bandwidth. Since QoS DiffServ is deployed end to end on the network, when load balancing occurs across multiple switches for a flow, the QoS, marks the packets, which make the reassembling easier, with minimal packet losses. During this process, the controller re-orders the links with the new forwarding table. When these occurs, an updates of newer flow table are sent across the network, such that the switches become aware of all changes ongoing in the network. Furthermore, the *controller* deploys SR ECMP on some of the active links selected to handle network flows, based on the percentage utilization threshold of the links. The *controller* generates newer segments and pushes these paths to the switches as well as a second backup link of the new backup segment, due to its global view of the network. Since the network is equidistant on the newly selected path (segments), ECMP is deployed on this selected SID alongside the SR-TE.

Then, in a situation when an oversubscription of a link occurs, i.e. if more traffic are generated which becomes more than the bandwidth available of active links, more switch utilization goes beyond the set S_{uTH} , which enables controller to trigger the deployment of a new spine switch on the network this process is similar to that explained in [1] [91]. During this situation, it is rare to see the spine switch utilization ratio to be less than 25% (please refer to Figures 12, 13, 15 and Table 3.3. The spine links are links connected between the spine switches of the network and leaf switches, while access links are links connecting servers, hosts to leaf switches). Hence, deploying the *power* module on such a network will be limited to monitoring the port utilization ratio, in other to reduce the spine links, instead of spine switches, for power efficiency. When any link is deactivated, congestion or bandwidth demand, the *controller* will either make use of the Link Aggregation Control Protocol (LACP) or fast reroute embedded in the SR header, for a quick link recovery, depending on which routing algorithm is been used. The Link aggregation control protocol is a way to bundle many different ports to form a single logical channel, it is very useful for load balancing links on a network [102]. As a result, the leaf-spine topology used in this thesis does not have the back-up link ring connected to the spine leafs core for fault tolerance as used in [94] and shown in Figure 2.5 in section 2.2.2.1. Although in practice this immediate backup ring link may be needed. But by creating an immediate failover during the network setup, and with the use LACP or segment routing (The LACP is an active controller API used southbound in the data plane by the *controller* for network recovery), this backup ring is not used in this system setup. Hence, with this system in place, more spine links and spine switches can be deactivated with a similar network performance, by so doing reducing the energy required to run the network.

The controller has APIs inbuilt, which enables it to derive needed When segment routing is been deployed over the network, and the controller, and the links are underutilized, the controller would be set to reduce the Active number of Activated Spine Switches (ANASS) on the network. To achieve these tasks, the initial load carried by each spine switch or link in question is rerouted to alternative links. Then the deactivation of the links or spine switches could be performed, without affecting the network performance nor packet losses. This is because the *controller* handles the changes with the routing algorithm in place. The *power* module can be used without the segment routing technique, and vice versa, as well as other modules that would be explained later in this chapter. All these modules can be integrated into the systems controller and dissociated from the system. The power module when deployed is developed in a way that, either the spine switches or spine links can be deactivated or activated, depending on the percentage utilization threshold of the spine switches, or ports on the switches. All these concepts are shown in the segment routing green energy management system were adapted from [1], as shown in Figure 3.1.

The following sub-sections expand on the modules used in the SR-GSSMS system.

3.2.1 The segment routing & routing module

The default routing algorithm in ODL used for routing between hosts through L2 switches is the Dijkstra's Shortest Path First (SPF) algorithm. Dijkstra's SPF algorithm is used in the iForwarding service API of the *controller* to the southbound switches. The SPF algorithm determines the shortest path between nodes utilized by the L2 learning switches [81]. The disadvantage of using this method is that each spine link connected to a leaf is

considered the shortest path to any other leaf connected by other spines. The reason is paths are all equidistant to the leaf switch, with a maximum hop of one. Thus, ECMP is deployed by default to all leaf nodes, as shown in the Figure 3.3 below. This makes the network the congestion prone as a result. Especially when a DCN handles different kinds of flows simultaneously, be it elephant flows and mice flows, when rearranging this flow across the spine switches and at the destination. One can conclude that the plain SPF algorithm of Dijkstra does not ensure the maximum utilization of network resources.

As shown in Figure 3.3 below, the spines are located in the middle, while leafs are on the sides of the spine. Figure 3 of Appendix A, depicts a network that is actually projected using the YANG API to visualize the nodes of the network. Despite that the available bandwidth on the spine links is 100 MB, because ECMP is deployed on the network, in sending 60 MB from node 1 to node 8 with legends in red, as shown in Figure 3.3, the flow will be split across all the spine links {15 MB across the spine links – link (1,3), link (1,4) link (1,5) link (1,6) and then reassembled at the destination port of node 8- link (3,8), link(4,8), link (5,8), link(6,8)}. A similar pattern is experienced when sending 1 MB simultaneously from node 2 to node 7, 250 kb is balanced across all the spine links {links (2,3), link (2,4), link (2,5), link(2,6) and then reassembled at the destination node 7 – link (3,7), link(4,7), link(5,7), link(6,7)} as shown in Figure 3.3 with legends in black. By balancing the load across the network this way, this makes all the spine switches active all the time, it also makes the network prone to congestion, and network failures. There will be no energy savings in this DCN, which is not energy efficient.

Figure 3.3 The default routing algorithm using ECMP on leaf spine Network

Looking at the network in Figure 3.3 above, every host is equidistant to each other. Hence inherently, ECMP is deployed by default on all nodes, because they have equal costs to the destination. Thus, the *controller* utilizes all spine links. The problem that arises in using this plain SPF routing algorithm is that when routing different sizes of flows, for example, a combination of elephant flows and short mice flows, congestion, bottleneck, high packet losses will occur, because when parsing elephant flows across spine links as well as mice flows, all flows are handled in a similar way as explained earlier. Furthermore, one can say that the network resources are not fully utilized. If a link in the network has a capacity of 100MB, and 1MB is to be sent across the network. In splitting this flow across the spine switches, this is not an effective way of utilizing network resources, because this flow (from node 2 to 7) mentioned above is subject or prone to network disturbances, as

well increasing the activity of spine switches and links across the network, unnecessarily. Furthermore, in down time period in DCN, all spine switches will still be kept active, because ECMP is to be deployed for all flows across the network, which is not energy efficient.

Using the network this way, results in an improper utilization of network resources in managing the network. Thus, a better routing algorithm is needed to efficiently utilize the network resources. Considering the traffic engineering explanation given in Chapter 2, we see segment routing is the best choice for traffic engineering in a DCN to be used, for an efficient utilization of network resources.

With the *segment routing* module been used, on start-up, the *controller* uses UNECMP segment routing. This ensures a utilization of minimum spine links in the network, thus enabling a higher number of spine links to be inactive or set to dormant. This module on network initialization, by default selects any shortest path using segment routing SIDs with the best available maximum link utilization (MLU) of an *active* link. Explanation of this algorithm is shown in Algorithm 1. The links selected become available and active, the others after a period of time may become inactive or dormant based on the utilization ratio of the spine ports. Using both node SIDs, and adjacent SIDs, depending on the situation as shown in Figure 3.4, Segment routing is deployed in this thesis to route flows across the DCN. Section 2.6 explains the concepts of the segment routing-TE deployed in Networks.

For simplicity in explaining these procedure, Figure 3.3 shows a simple fraction of the entire network, before further segment routing is deployed. The spines are located in the middle while leafs are on the sides of the spines. (Please see Figure 3 in Appendix A).

Figure 3.4 The modelled nodal-SR on leaf-spine DCN

In Figure 3.4, we see more links which are not in use compared to links which are been used, for the same network flows as used in Figure 3.3. The links that are dormant are those on designated by the controller as back up links, as explained earlier in this chapter. The links in use are colored, and are shown with the legend accordingly. Since the flow of 60 MB is not over the link capacity, of node 1 through 6 to node 8, the forwarding table for leaf 1 would be as shown in table 3.1.

Table 3.1 The forwarding table for node 1: an example

Leaf 1	Out (SID)	Out_Intf
Base ID 100	<1603>	To node 6

To further explain the concept of segment routing for auto-bandwidth management, we create another scenario on the same network. As depicted in Figure 3.5 for the same network, the curved lines represent the segments with IDs, while the black lines are links not used the coloured lines have their legends with BW. Initially, the segment ID 1601 is used for the 1Mb of flow from leaf node 2 through node 5 then to 7. As a result, there is still available bandwidth on the active spine switch at node 5. The link speed of the entire network is approximately 100 Mbps. As seen above if 60 MB is transferred from node 1 through 6 to node 8, since it's not beyond the link capacity of the link (100MB) nor the configured threshold at 80MB, no action is done, except for forwarding the flow. These leaves the forwarding table for node 1 to the same as shown in Table 3.1.

Figure 3.5 The modelled-SR policy with auto-bandwidth across active segments

Table 3.2 The new forwarding table for node 1

Leaf 1	Out(SID)	Out_Intf
Base ID 100	<1602, 1604>	To Node 5
	<1603>	To node 6

But, if the flow from node1 to node 8 increases changes to 150MB elephant flow this is more than the MLU of the spine links, which is approximately 100Mbps. This is where auto-bandwidth, comes into action, the leaf switch communicates with the controller for actions to take as explained earlier. The network automatically activates the back-up links and uses it for load balancing the excess Bandwidth across the links. This change is relayed to the controller, which creates a new segment to be reserved for backup or failover link

for the flow. In load-balancing, the flow is equally balanced across the backup link and the initial active link.

A new spine switch would not be considered, unless the Maximum utilization threshold of 85 % (the S_{uT-Max}) is reached, or the switch utilization threshold high of 75% (the S_{uT-H}) is reached for set period of time T (table 3.3 for a list of notations, please refer to algorithm 4 and 5, the value of the notations used is dependent on the network admin), unless it is needed for the new flow i.e. there is no other link to the destination node. Instead an existing switch 5, will be considered for routing. Thus, utilizing segments 1603 and both node SID 1602 and adjacent SID to node 5 (SID 1604) are utilized. As a result, 2 segments and 3 SIDs are used for forwarding these flows. The new forwarding table for node 1, changes as shown in the table 3.3. This forwarding table shows the changes from the default segment 1603 in output to a combination of the SIDs.

Since some packets are separated and sent over different spine switches, in reassembling the same packet at the destined leaf switch, QoS is also needed in this special instance, thus the need of a scalable end-to-end QoS which is initially deployed across the network, as mentioned in section 3.1.1. This is because apart from splitting a flow, the assembling of the same flow is very vital, so as to ensure proper packet delivery and minimal packet loss. QoS (DiffServ as explained in section 2.5.5 and 3.1.1), is used, to ensure the packets are assembled in the destination in the proper order. This is because DiffServ helps in packet marking, so the reassembly of the same packet in the destination is achieved properly, this reduces the packet loss as a result. Despite QoS is used in Figure 3.3 scenario, we still see large packet losses due to congestion poor packet handling with

different flow sizes. But with SR, the packet loss is reduced drastically due to proper packet handling across the network, as shown in Figure 3.5. This results in ease in deploying energy efficiency algorithm across the network.

3.3 Algorithms

This section provides details of all the algorithms used in this work. Setting out all the details of the traffic engineering integrated with power algorithms deployed in the SR-GSSMS systems as explained below, since the default routing algorithms have been discussed above, we would just focus on the algorithms used for this thesis work.

3.3.1 Algorithm 1: Segment routing with auto-bandwidth.

This algorithm can be explained as follows: the *controller* gets a complete picture of all nodes, links, and bandwidths available on the links, when the network is setup. With communications from the network with API's the *controller* then calculates based on the network resources the best routes to each switch and dumps the forwarding table for each switch. The controller is set in such a way to deploy UNECMP on the spine links, such that for each flow or destination in the forwarding table, there will be two links, one is active and the other is set redundant (or passive) which becomes the failover link. The auto-bandwidth with segment routing explained above is not deployed initially, unless warranted. But, if the controller uses the default SPF routing algorithm, it results in higher latency and congestion, when different sizes of flow are being handled, as explained earlier in in this chapter.

Furthermore, after the controller dumps the flows of segments to the switches proactively, and selected backup links. If any flows bandwidth is below the set threshold available, the switch node simply forwards the request to its destination. If not, the switch uses the backup link immediately and balances the flow across the backup link. It then relates this action through APIs to the controller on the state and then the controller proactively uses create a new backup segment ECMP load balancing is deployed over the 2 links, to handle the higher bandwidth, and the backup link is in place should in case this is not sufficient for the flow. The links are aggregated and updates of forwarding table is broadcasted across the network. This algorithm can be written as follows:

Algorithm 1: SR with auto-bandwidth

Initialization: Controller	
1.	Collect information of links, nodes, bandwidth, and their connections in the network
2.	Calculate/choose any shortest paths/segments to each node for all switches
3.	Create segments, and backup links for nodes and adjacent IDs
4.	Dump segments to forwarding table with segments IDs to all switches
5.	If (a packet destination is reachable by a link on datacenter network)
6.	If (link path BW is sufficient for the flow)
7.	Forward the flow to destination segment
8.	Update flow table
9.	Else
10.	Use alternative segment(s) for the excess bandwidth
11.	Load balance the excess flow over the two links
12.	Forward the flow to destination using all link paths
13.	Use API's to inform controller of changes
14.	Controller choses a second backup link for the network
15.	Controller makes link reservation
16.	Re-order the links and bandwidths
17.	Update flow table
18.	End if
19.	Else
20.	Delete request, delete link reservation
21.	Delete flow update from the table
22.	End if

Table 3.3 shows the list of parameters used for the SR-GSSMS. Each with an explanation beside.

Table 3.3 List of Notations (adapted from [1])

Components	Controller Parameter (Module)
B_{ij}	Total number active spine links
D (i,j)	List of all link on network from (i, 1, 2, 3, j)
G	Number of all Spine switches used
S_{tA}	Spine switch to be activated
S_{tOFF}	Switches to be turned OFF
S_{tON}	Switches to be turned ON
S_{uT-H}	Switch utilization Threshold high
S_{uT-Max}	Switch utilization Threshold maximum
S_{uT-Min}	Switch utilization Threshold minimum
S_{uT-L}	Switch utilization Threshold low
T_{addL}	Time duration to deactivate link
T_{addS}	Time duration to deactivate switch
L_{tOFF}	Links to be turned OFF
L_{tON}	Links to be turned ON

3.3.2 The power control module and algorithms with SR

The power control module consists of algorithm 2 and algorithm 3 as explained below. Algorithm 2 is designed to reduce the ANASS, whereas algorithm 3 is designed to reduce the spine links by monitoring the port statistics on the controller. The power control module could be deployed using either algorithm 2 or algorithm 3 or a combination of both algorithms, depending on the network engineer. In cases of very high oversubscription ratio like 3:1, it is rare to make use of algorithm 2. This is because the spine switch utilization ratio will be always higher than S_{uT-L} . But if the subscription ratio between the access link and spine link is 1:1, algorithm 2 or algorithm 3 or a combination of both, can be deployed on the network. In this thesis, we made use of over subscription ratio of 1:1 in the network, as a result the explanation covers both algorithms 2 and 3.

Figure 3.6 below shows the concept of the power algorithm. These algorithms is deployed on the spine links because of the UNECMP, been used in the network. As a result, there are more redundant spine links available on the network which can be reduced to improve the energy efficiency of the system. From Figure 3.6 below, we can see a simple leaf-spine DCN with 3 active spines and 6 leafs with two back up spines switches. Of the active spine switches, one spine switch has only one active link. Since SR is deployed, the less active load on this spine switch can be transferred to that of others, without really affecting the network functions, or metrics. During this period this switch becomes the switch to be turn off (St_{OFF}), hence it is the *controller* uses the network *monitor module* to monitor the St_{OFF} . This makes the switch utilization ratio of the St_{OFF} , to go below the set value of S_{uT-L} , for a set period of time T. The controller then deactivates this St_{OFF} . On the other hand, there are passive switches ready for activation, if the reverse is the case.

Figure 3.6 Active and inactive links and spine switches (adapted from [1])

In simple terms, the first part of the power algorithm for deactivating spine switches comes active when a spine switch utilization goes below the S_{uT-L} . Then a timer T is turned on. If the situation remains then, the controller transfers all the network functions and flows of the switch S_{OFF} to another switch without affecting the network behaviour significantly because segment routing is used. The switch S_{OFF} becomes deactivated, the controller updates the forwarding table, and aggregates the network. Otherwise the timer is deactivated and the network remains the same.

3.3.2.1 Algorithm 2: Decreasing active spine switches

Decreasing the number of active spine switches	
1.	G = The number of active spine switches
2.	Deploy UNECMP-SR on spine switches
3.	Monitor the switch utilization of spine switches
4.	Select the least utilized spine switch and make it St_{OFF} .
5.	For all active spine switch G
6.	If the switch St_{OFF} flow utilization of the connected spines is $< S_{uT-L}$
7.	Activate the timer T
8.	While $T < T_{dds}$
9.	If switch St_{OFF} utilization $> S_{uT-L}$
10.	Deactivate timer T
11.	Else if $T > T_{dds}$
12.	Use SR to route all flows on the St_{OFF} to neighboring switches
13.	Deactivate the switch St_{OFF}
14.	End If
15.	End While
16.	Aggregate the links
17.	Update the forwarding table
18.	Let the network continue as it is.
19.	End if
20.	End for

A similar process is done when a link is deactivated instead of a switch, as seen in algorithm 3. Algorithm 2 describes the actions the controller takes. The controller observes the flow utilization of all spines switches, compares the percentage to the configured value, S_{uT-L} , which is 25% of switch utilization. When a switch utilization is much lower than others, it becomes the St_{OFF} . The controller continues to monitor its utilization percentage when it becomes lower than 25%, the timer T becomes activated. If during this process, the switch utilization becomes higher than 25%, the timer is deactivated. If not, the controller waits on the timer, when time T is greater than the time to deactivate switch (T_{dds}), the controller transfers the function of the switch to other spine switches; thereafter, it deactivates this switch. It must be mentioned that, if the S_{uT-L} is lower than the least

Switch utilization ratio minimum (S_{uT-Min}), the controller would not activate a timer. Instead, an immediate action will be taken, as in this situation a spine switch or multiple spine links could be deactivated, if the network engineer is focusing first on the links. This is done so as to handle switches with low utilization experienced in the network.

3.3.2.2 Algorithm 3: Decreasing active spine links

Decreasing the number of active spine links	
1.	B = The number of active spine links connected
2.	Deploy UNECMP-SR on spine links
3.	Monitor the port utilization of spine switches for the links
4.	Select the least utilized spine link (port) and make it L_{tOFF} .
5.	For each link between spine and leaf switch $l_{ij} \in B_{ij} \in D_{ij}$
6.	If the switch S_{tOFF} flow utilization of the connected spine is $< S_{uT-L}$
7.	Activate the timer T
8.	While $T < T_{dds}$
9.	If switch S_{tOFF} utilization $> S_{uT-L}$
10.	Deactivate timer T
11.	Else if $T > T_{dds}$
12.	Use SR to route the flow on the L_{tOFF} to neighboring switches
13.	Deactivate the link L_{tOFF}
14.	End If
15.	End While
16.	Aggregate all the links
17.	Let the network continue as it is.
18.	End if
19.	End for

Algorithm 3 is similar to algorithm 2, except algorithm 3 is used to deactivate links, instead of spine switches. It should be noted it is better to deactivate spine links one at a time based on the VMs handled on leaf switches connected to the spine link, because the number of servers connected to a spine link cannot be compared to that of a spine switch which has multiple spine links spread out. The link recovery time in deactivating a single link is almost instantaneous, unlike if multiple spine links are deactivated simultaneously.

The process of link aggregation or fast recovery on re-route is much easier for a single link, one at a time as compared to many flows handled by a switch.

Algorithm 4 and algorithm 5 functions the opposite way as to algorithms 2 and 3. Algorithm 4, increases the number of spine switches, with backup switches as shown in Figure 3.5. During a peak bandwidth demand, the switch utilization usually increases above 75%, over time backup switches becomes activated to compensate and augment the network bandwidth demand. These backup switches St_{ON} are set in place and connected such that when in demand, the backup spine switches become activated to balance the bandwidth demand in the network, and with segment routing been used, the load transfer is made to work almost seamlessly. As explained for algorithms 2 and 3, if the S_{uT-H} is higher than the S_{uT-Max} , the controller would not activate a timer, instead an immediate action will be taken, a spine switch would be activated or multiple links. This is done so as to handle the high switch utilization experienced in the network.

3.3.2.3 Algorithm 4: Increasing active switches

Increasing the number of active switches	
1.	G = The number of active spine switches
2.	Deploy UNECMP-segment routing on spine switches
3.	Monitor the switch utilization of spine switches
4.	Select a dormant spine switch and make it St_{ON} .
5.	For all active spine switch G
6.	If any spine St_A switch flow utilization is $> S_{uT-H}$
7.	Activate the timer T
8.	While $T > T_{das}$
9.	If switch St_A utilization $< S_{uT-H}$
10.	Deactivate timer T
11.	Else If $T > T_{das}$
12.	Activate the switch St_{ON}
13.	Use SR to route balance the network and route flows to the St_{ON}
14.	End If

15.	End While
16.	Update the forwarding table
17.	Let the network continue as it is.
18.	End If
19.	End For

When the StA utilization becomes more than S_{uT-H} for a period of time T . First, if there are any dormant links of active switches, this will become active before, activating a new spine switch. In doing this, since segment routing is been used, the controllers auto-bandwidth automatically balances the load, on the links newly activated link. This is done this way because the number of links that each spine switch connects to, is same as the number of leaf switches available in the network. Thus, it takes a little more time to balances all traffic when a spine switch is activated as compared to first using the dormant spine links, before turning on spine switch.

3.3.2.4 Algorithm 5: Increasing active links

Increasing the number of active links

1. G = The number of active spine switches 2. Deploy UNECMP-SR on spine switches 3. Monitor the switch utilization of spine switches 4. Select the link to be activated and make it L_{tON} 5. For each link between spine and leaf switch $l_{ij} \in B_{ij} \in D_{ij}$ 6. If any spine StA switch flow utilization is $> S_{uT-H}$ 7. Activate the timer T 8. While $T > T_{das}$ 9. If switch StA utilization $< S_{uT-H}$ 10. Deactivate timer T 11. Else if $T > T_{das}$ 12. Activate the spine link L_{tON} 13. Use segment routing to balance the network & route flows to the L_{tON} 14. End If 15. End While 16. Aggregate the links 17. Let the network continue as it is.
--


```
18.   End if  
19. End for
```

As explained, the proposed algorithms in the *power* module work properly in a DCN with leaf-spine topology, to make up the SR-GSSMS. When segment routing is deployed, it makes the changes occurring in the network not to affect the performance of the network, unlike the case where segment routing is not been deployed. Chapter 5 presents the experimental results to show how advantageous it is to use segment routing over the traditional SPF routing. The proposed approach also gives the network the needed resilience and flexibility to deploy the SR-GSSMS power algorithms.

Chapter 4: Simulation Setup and Results

The simulation environment used for this research is the ODL controller for SDN. This network environment has the limitation of representing VMs. We also assume that VM are used as servers, which are deployed in the hosts connected to the leaf switches. In this environment, we not able to transmit traffic using sin waves or any trigonometric waves as used in [1]. Instead, the focus is placed on the network performance, as to maximally utilize network resources, so as to provide some energy savings in the process as explained in chapter 3. Hence, a measure of energy efficiency, packet loss, BW utilization, and jitter are criteria used to evaluate the significance of the proposed SR-GSSMS.

In this thesis, the simulation was carried out using Ubuntu as the operating system and the Intel Core i7-3770 CPU@2.40GHz $\times$ 8 as the processor. Mininet [70] [71], was used to simulate the network on ODL controller, with the hosts and servers connected to the access layer to the leaf switches which are OVS switches. The network was set up so as to imitate the conventional rule of thumb in building a leaf-spine DCN, which is an oversubscription ratio of spine link speed to access link speed of 1:1 [4] [57]. The most important limitation shown in Mininet documentation is that it currently has a threshold of 1 Gbps link speed and a port speed of up to 10 Gbps for good performance [70] [71]. Therefore, as shown in Table 4.1, the experiments were carried out using **100 Mbps** for access port (to represent a **10 Gbps** link conventionally used) and **100 Mbps** for spine leaf links (to represent the **10 Gbps** conventionally used for leaf-spine links a conventional standard used and recommended by both Cisco and Arista in building leaf-spine DCNs [4] [57]).

Since the focus is to study the leaf-spine topology, as to network architecture performance, the topology changes were not resonated to the hosts, to test performance as to different oversubscription ratios. For simplicity, we used a fan-out of 1 and in some experiments a fan-out of 4 was used which generated similar results (i.e. the number of host connected to a leaf switch was 1, and in some experiments 4). The networks we tried to simulate, are adopted from Cisco practices [91]. For most of the experiments we had a constant number of leaf switches of 192, except otherwise stated, while the number of spines, as shown in Table 4.1, was varied for various test cases. We ensured that for all condition, even when experimenting a percentage utilization of servers as low as 30%, the minimum number of spine switches that must be active is at least two. Other parameters used for experiments and the value(s) used for each parameter are presented in Table 4.1. One of Figure for the network used is shown in Appendix A Figure 3, using YANG API, and DLUX core API functions and MDSAL, in SDN.

The evaluation metrics used in the simulation are listed as follows. The first two metrics were adopted from [1] [97]:

- Average Number of Active Spine Switches (ANASS)
- Percentage of Power Consumed (POPC) over the total number of Spine switches.
- Percentage of Lost Packets (PLP).

PLP is calculated in place of the Percentage of Failure (PF) as used in [1] [97]. This is because the simulation results only show the number of packets dropped. This occurs

more often when SR redirects a flow from one switch link to another. ANASS, POPC, and PLP are calculated with equation (4.1), equation (4.2) and equation (4.3), respectively. In equation 4.3, N is the total number of spine switches. For power savings, it is assumed that a switch placed in hibernation mode can lead to a significant power savings, i.e., 84%, as reported in [97]. Therefore, in equation (4.2), a value of 0.16 is used, (i.e., $1 - 0.84$, is used to calculate POPC).

Table 4.1 List of experimental parameters and values used

Network parameters	Value(s) used
Entire Network Bandwidth (Iperf average)	97.5 Mbps
No of leafs	60, 96, 148, 192
No of spines	2, 4, 6, 8, 10, 12, 14, 16
Fan-out	1,2, 4
Long Flow size (UDP)	80, 100, 120, 150 MB
Short flow size (UDP)	2, 10, 20 MB
Link BW between access port (leaf-hosts)	100 MB
Link BW between spine port (leaf-spine)	100 MB

$$ANASS = \frac{\sum(\text{number of active Spine switches} \times \text{time})}{\text{Total time}} \quad (4.1)$$

$$POPC = \frac{ANASS + (N - ANASS) \times 0.16}{N} \times 100\% \quad (4.2)$$

$$PLP = \frac{\text{the number of packets lost}}{\text{the number of all flows}} \quad (4.3)$$

In carrying out the experiments, uniform traffic pattern was used to simulate real traffic in a network. We made use of the user datagram protocol (UDP), because although the

packet delivery can be monitored and measured, it does not ensure packet delivery. As a result, the difference in packet received from sent becomes the packet loss measured over the network. This is the value of the PLP in the network for a flow. Unlike TCP, which ensures packet delivery, by resending the lost packet. UDP was used to test the performance of the networks for this paperwork.

4.1 Energy Savings with SR-GSSMS in DCN

The experiments carried out were used to imitate the traffic pattern in a DCN. Looking at the energy charts and server utilization of data centers from the Dell and Movri [18] [19], we see that a DCN is usually busy when its consumers, the users it serves, are active and working. The DCN would also have some certain patterns of network utilization. Which is similar and can be derived from the charts shown in [18] [19]. These patterns were emulated in the experiments. There are periods when the server's utilization would peak up, we represent this as 80 % utilization of servers in the network, and when the server's utilization will be low we represent this as 40% of server's resources in the DCN. The result of these traffic patterns on the DCN is shown in Figure 4.1 and Figure 4.2. Figure 4.1 shows that the POPC, a factor of the percentage of server's utilization for the leaf switches, the higher this percentage the higher the POPC. When the network utilization is 40%, the POPC is went down as low as 32.5%, which means an energy savings of up to 67.5%. This shows the higher the network utilization, the higher the POPC. Thus, we can see from the Figure that energy savings in the network is of the range of 11.1% to 67.5% for a 90% utilization and 40% utilization, respectively. The proposed SR-GSSMS can save even more energy, should the network utilization of servers' usage be 30% or lower.

In the experiment, in representing the server's utilization, we ensured to make use of a uniform traffic UDP, randomly distributed over the network as a whole. We made use of elephant flows of 150 Mbps, 120 Mbps, 80 Mbps, 40 Mbps 12 Mbps and 2 Mbps randomly. This is to represent the constant varying traffic seen across a DCN. Figure 4.2 shows a small increase in the PLP on the network. This can be attributed to the changes in flows, which is handled by the segment routing operation in the network. It also shows a decrease in the packet loss as there is an increase in the percentage of active servers. This is attributed to the increase in the ANASS.

Figure 4.1 POPC performance results

Figure 4.2 PLP Performance results

On a normal network in which SR-GSSMS is not deployed, the ANASS will be fixed at 16, for all the varying traffic and percentage of servers usages. Be it 80% or 40%. All the DCN spine links and switches will be always set on. This is not an efficient way to utilize the spine switches in terms of energy. In some DCN today, we may see many spine switches are in use to serve only a few servers in the leaf switches. Using the DCN this way does not save energy.

To maximize the benefits of the proposed SR-GSSMS, we make use of the migrating VM technology between leaf switches, such that leaf switches which have less activities could be turned to sleep, first by transferring the VMs of servers from leaf switch with a low usage to that with higher usage, this is later reflected on the spine links, where the

functions of the spine switches with low utilization threshold, are transferred to spine switches with higher utilization threshold [1]. In these experiments, we represent VM migration as a function of servers in hosts connected to leaf switches due to the experimental environment.

4.2 Comparison between SR-GSSMS and the default routing technique

In this comparison, we took into consideration various network parameters, which would be highlighted in this section. We also ensured a similar network is deployed on both network systems with segment routing and without segment routing. In these experiments, the results shown represent an average of 5 different experiments carried out for the network, except mentioned otherwise.

4.2.1 Comparison of PLP on SR-GSSMS and the default routing system.

For this section of experiments, we tried to emulate the power module, as depicted in Figure 3.6, of saving spine switches on the networks of both systems. This was done to show the importance of segment routing on the leaf-spine DCN. When segment routing is deployed on the network, we see an insignificant packet loss on the network. The power module as used in this experiment was induced to test and compare the network performance, when the system runs without segment routing. While we maintained a constant number of leafs on the entire network of 96, the number of spines in these experiments were varied accordingly from 12, 10, 8, 6, 4 as shown in the table 4.1.

During this experiment, using the default SPF routing algorithm, as shown in Figure 4.3 (NO SR), the percentage packet loss rose rapidly, up to an average of 60% PLP with a

decrease in the number of spines, for the same amount of UDP packets transferred over the network topology. Hence, both congestion and a higher number of over-utilized spine switches occur as a result, as can be seen in most DCNs. On the other hand, when the segment routing is deployed for the same experiment, the percentage packet loss is insignificant. As shown in Figure 4.3 (with SR), the controller properly re-allocates spine links resources especially when there is a reduction in the number of spine switches. This is to maximally utilize the available BW on the spine network (96GB – 307 GB). It is because of the network results shown in Figure 4.3, which makes it possible to easily increase or reduce the ANASS in the network, whilst maintaining the network performance. Thus, one can say the SR-GSSMS is energy efficient, because it increases the possibility of reducing the ANASS and increasing the efficiency of the utilized energy in the leaf-spine DCN.

Figure 4.3 Comparing PLP with and without segment routing

The standard deviation was calculated from an average of 5 experiments for each node. Considering each network, the standard deviation of each network ranges from 0.0053 to 0.4 for the networks with SR, whilst networks without SR, standard deviation ranges, from 0.816 to 2.5.

It should be noted that, the results without SRGSSMS shows a very large packet loss, this is because elephant flows of UDP were being used, hence, a delay in response, of a few seconds' results in an enormous packet loss. There were a few cases in the experiments where the systems with SRGSSMS enabled responds late to anomalies such as congestion or link failures, this results in packet loss of about 5%, in very few cases about 0.5% of the time for some of the experiments. Hence this rare cases were not shown on the plot as well as in other plots similar to this. But in general the system responds quickly to changes.

4.2.2 Performance results with varying packets sizes

To measure the network performance as to packet size, a varying size of UDP packets were transmitted simultaneously, over the network. So as to measure the performance of the network, relative to the size of the UDP flow. Figure 4.4, shows a difference in the performance when the same packet size is transmitted over the network. The value plotted on the Y-axis, is a combination of both the packet size sent and the percentage loss. The spine switches, was kept same, at 8 spines. So as to actually measure the network performance with variation in the packet flow over the network.

Representing the PLP performance results for the default network, without segment routing is Figure 4.4(a). This shows an insignificant PLP for cases 1-3. But for cases 4-7,

we see a rapid increase in the packet loss of up to an average of 84%. This can be attributed to the packet size, the link bandwidth and how the controller sends packets by default. As explained in Chapter 3, the controller uses ECMP by default, when ECMP is deployed on all packets, transmitted simultaneously, this usually results in congestion, and bottleneck in the network. As a result, we see a fast increase in the PLP across the network. We can conclude that, the higher the packet size, the higher the PLP on the default network.

On the other hand, Figure 4.4(b) shows an insignificant PLP for cases 1-5 (with SR). This is because the controller pro-actively manages the flow from each switch with the purpose of fully utilizing the available large bandwidth between the leaf and the spine switches. It was noted that there was some packet loss, due to the large size of flow coming from many leaf switches in the network simultaneously or can be attributed to the controller changing the flow direction in the network; during this process, the packet losses increase but remain insignificant, which is shown to the Figure 4.4(b). Since the packet size is within the bandwidth of the spine network layer, the controller properly handles the flow, making the packet loss to be insignificant. As a result, the standard deviation of these experiment with SR was in the range of 0.00074-0.44, while the standard deviation without SR, was in the range of 2.06-5.5.

(a)

(b)

Figure 4.4 Comparing PLP of network compared to different UDP transmitted. (a) without segment routing

(b) with segment routing

4.2.3 Time taken to connect the network to the controller

Figure 4.5 below shows a similarity in the time taken for the Mininet network to connect to the controller for both default routing and SR. The reason is that it is the same network being used, hence a very similar amount of time is required to connect the network to the controller. The results show values for with segment routing and without segment routing on the controller. This experiment was carried to compare the effect of routing algorithms to the controller. The X-axis is for the number of spines switches, while the Y-axis is for the time taken to connect to the controller measured in seconds. The Figure 4.5 shows a similarity as to the time it takes to connect the network to the controller. Hence, we can conclude that the routing algorithm has no effect on the time it takes to connect to the controller. Instead, it is the number of devices on the network that affects the parameter. The higher the number of spine switches, the longer it takes to connect to the controller.

Figure 4.5 The time taken to connect to the controller

4.2.4 Measurement of latency

To calculate average latency over the network topology, various Internet Control Message Protocol (ICMP) packets were used from different hosts to ping, other hosts randomly and simultaneously. It should be noted, if these experiments were performed with the network not busy with traffic, the results will be almost similar. It will be a comparison of the default SPF algorithm to segment routing algorithm. But since we are focusing on network latency measurement simulation, multiple ping flows are used simultaneously as in the previous experiments. This creates traffic over the network, hence we can measure the true latency of the network, comparing the latency of using the default routing algorithm to SR algorithm. In the experiments, 100 ICMP packets were used for each node to ping other node in order to test the latency on the network. This is because, the latency result for each transmission of one host to another host varies, an average of the latency experiments for each host was first computed. Then to calculate the average latency of the entire network, an average of the different averages of latency from different host was computed to be the average value of the ICMP latency test over each leaf-spine DCN for each case.

The results in Figure 4.6 shows an insignificant increase in the latency when segment routing was used, compared to when segment routing algorithm was not used. This large difference can be attributed to the delay in traffic created in a network which does not use SR in rearranging the packet flow at the destination, because ECMP is deployed by default. Unlike when the SR routing algorithm is deployed, ECMP is avoided on all spine links unless needed.

As explained earlier in section 3.1 of this thesis, the controller dumps the best segment to route traffic for each leaf switch. As a result, it not only uses the best path, but also there is a minimum delay. Since the jitter is a measure of variations in latency, the experimental readings of jitter are not computed. The experiments actually revealed a similar result, as can be seen in some of the images posted in the appendices.

Figure 4.6 Measure of latency across the network.

4.3 The effect of varying some switch metric parameters on the network

Figure 4.7 shows the effect of varying the switch utilization threshold-High, i.e., S_{uT-H} , on ANASS. In this Figure, each bar legend represents a set value of the S_{uT-H} . The Figure 4.7 shows that by decreasing the S_{uT-H} the ANASS increases slightly. Such that for the same percentage utilization of the servers in the network, the ANASS for S_{uT-H} is generally

highest for the least S_{uT-H} . The X-axis shows the percentage number of active servers, while the Y-axis shows the number of ANASS activated for each case of S_{uT-H} . The varying legend represents each value of S_{uT-H} which is shown for the varying number of active servers deployed on the network.

Figure 4.7 The effect of varying the S_{uT-H} on the ANASS.

Figure 4.7 above is shown to emphasize the effect of the parameter S_{uT-H} on ANASS. It is such that, when the S_{uT-H} of the spine switches is lowered, this results in a slight increase on the rate of triggering an increase in the number of spine switches, because this threshold is attained more quickly, compared to when the S_{uT-H} is higher. Since the S_{uT-H} is the threshold used to trigger the activation of spine switches. So, when it is low, this increases the possibility of deploying more switches and vices versa. This is because the threshold is reached quickly, hence more ANASS are deployed.

On the other hand, we see a similar trend in Figure 4.8 when the switch utilization Threshold-Low S_{uT-L} increases, we notice there is a slight decrease in the number of ANASS. Such that for the same percentage utilization of the servers in the network, the ANASS for S_{uT-L} is usually lowest for the highest S_{uT-L} . The X-axis shows the percentage number of active servers, while the Y-axis shows the number of ANASS activated for each case of S_{uT-L} . The varying legend represents each value of S_{uT-L} which is shown for the varying number of active servers deployed on the network.

Figure 4.8 The effect of varying the S_{uT-L} on the ANASS

Figure 4.8 is shown to emphasize the effect of the metric parameter S_{uT-L} (meaning of the parameter) on the network. When the S_{uT-L} of the spines switches reduces, this results

in a slight decrease on the rate of triggering the shutdown on number of spine switches, hence the active switches continues to remain active despite there is a reduction in activity. Thus, we can simple say that a decrease in the S_{uT-L} results in an increase in the ANASS, and an increase in the S_{uT-L} results in a decrease in the ANASS, as shown in Figure 4.8 above.

In conclusion, the proposed SR-GSSMS is able to achieve a POPC of almost 30% as shown in Figure 4.1, which is an energy savings of up to 78% based on ANASS This value can be further increased should in case the percentage of active servers are reduced below 40%, (For clarity purpose, the value of the energy savings 78%, is not a measurements of hardware energy measurements, Instead, in actual sense, it is a measure of the energy that will be saved based on the actual energy consumption of the switches in each any DCN, hence it will be an energy savings of 78% of the total energy consumption of the networks in a spine-leaf DCN, when this algorithm is used).

This can be attributed to the power algorithms working in hand with segment routing to for the SR-GSSMS. In measuring the performance, we have shown a much better performance of the network metrics as to network latency, packet losses, and jitter, when segment routing is deployed compared to when it is not used on the DCN. We have also seen how the switch metric threshold values affect the ANASS in the network, showing the relevance of these parameters in energy savings as well, be it for the Switch utilization threshold-High (S_{uTH}) or the Switch utilization threshold-Low (S_{uTL}).

Chapter 5: Conclusion and Future works

In conclusion, we started by highlighting the importance of our choice of DCN, the leaf-spine topology. We also highlighted the importance of using segment routing technology, in traffic engineering. Furthermore, we have been able to show that with proper management of spine switches and links, using the proposed SR-GSSMS with focus on auto-bandwidth, gives much better results, combined with a flexibility of usage. When been properly used, it often results in a well-controlled traffic engineered network, with minimal packet loss. When properly managed, the results the SR-GSSMS yielded up to 78% of energy savings, whilst maintaining the almost equal network performance metrics.

We recommend this proposed SR-GSSMS technology to many DCN, as using this technology properly will yield not only good traffic engineering, energy savings, but will also improve the performance of their DCN in terms of performance as to networking issues, which often occurs.

5.1 Future works

It can be said that further research would be needed to test the performance of the SR-GSSMS algorithm on an NFV paradigm, also, for to properly test this algorithm in a hybrid network of SD-WAN. It is recommended, to properly measure the network performance of the DCN as different oversubscription ratio 1:1, 2:1, 3:1, 4:1, and 5:1, from access to core over the leaf-spine topology. So as to properly determine how to overcome oversubscription ratio issues facing DCN.

References

- [1] Li, X., Lung, C.H. and Majumdar, S. “Green spine switch management for datacenter networks”. *Journal of Cloud Computing*, 5(1), 2016, pp.1-19.
- [2] AWS & Sustainability. 2017. [Online]. Available: <https://aws.amazon.com/about-aws/sustainability/>. [Accessed: 10- Feb- 2017].
- [3] Lebednik, B., Mangal, A. and Tiwari, N., May 2016. “A Survey and Evaluation of Data Center Network Topologies”. arxiv.org.
- [4] Cisco. “Massively Scalable Data Center (MSDC) Design and Implementation Guide.” Oct. 2014, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/MSDC/1-0/MSDC_AAG_1.pdf, [Last Accessed on 14- Jan- 2017]
- [5] J. Shuja, K. Bilal, S. Madani, M. Othman, R. Ranjan, P. Balaji and S. Khan, "Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers", *IEEE Systems Journal*, 10(2), 2016, pp. 507-519.
- [6] Kant, K. “Data center evolution: A tutorial on state of the art, issues, and challenges.” *Computer Networks*, 53(17), 2009, pp.2939-2965.
- [7] Bergamaschi, R.A., Piga, L., Rigo, S., Azevedo, R. and Araújo, G. “Data center power and performance optimization through global selection of p-states and utilization rates”. *Sustainable Computing: Informatics and Systems*, 2(4), 2012, pp.198-208.
- [8] R. V. Aroca, L. M. G. Gonçalves, "Towards green data centers: A comparison of x86 and arm architectures power efficiency", *Journal of Parallel Distributed Computing*. 72(12), 2012, pp. 1770-1780.
- [9] Li, S., Lim, K., Faraboschi, P., Chang, J., Ranganathan, P. and Jouppi, N.P., 2011, December. “System-level integrated server architectures for scale-out datacenters”. In *Proceedings of IEEE/ACM 44th Annual International Symposium on Microarchitecture* (pp. 260-271).
- [10] Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T. and Reinhardt. “Understanding and designing new server architectures for emerging warehouse-computing environments”. In *Proceedings of IEEE/ACM SIGARCH on Computer Architecture News* 36(3), 2008, pp. 315-326.

- [11] Keys, L., Rivoire, S. and Davis, J.D., 2010, June. "The search for energy-efficient building blocks for the data center". In *Proceedings of Springer's International Symposium on Computer Architecture* (pp. 172-182).
- [12] Wang, Y., Liu, D., Qin, Z. and Shao, Z. "Optimally removing intercore communication overhead for streaming applications on MPSoCs". In *IEEE Transactions on Computers*, 62(2), 2013, pp.336-350.
- [13] Xu, J. and Fortes, J.A., 2010, December. "Multi-objective virtual machine placement in virtualized data center environments". In *Proceedings of the IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing* (pp. 179-188).
- [14] Kgil, T., Saidi, A., Binkert, N., Reinhardt, S., Flautner, K. and Mudge, T. "PicoServer: Using 3D stacking technology to build energy efficient servers". In *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(4), 2008 p.16.
- [15] Pelley, S., Meisner, D., Zandevakili, P., Wenisch, T.F. and Underwood, J. March. "Power routing: dynamic power provisioning in the data center". In *ACM Sigplan Notices* 45(3), 2010, pp. 231-242.
- [16] Floyd, M.S., Ghiasi, S., Keller, T.W., Rajamani, K., Rawson, F.L., Rubio, J.C. and Ware, M.S. "System power management support in the IBM POWER6 microprocessor". *IBM Journal of Research and Development*, 51(6), 2007, pp.733-746.
- [17] Simanjorang, R., Yamaguchi, H., Ohashi, H., Nakao, K., Ninomiya, T., Abe, S., Kaga, M. and Fukui, A., 2011, March. "High-efficiency high-power dc-dc converter for energy and space saving of power-supply system in a data center". In *IEEE 26th Annual Exposition of Applied Power Electronics Conference (APEC)*, (pp. 600-605).
- [18] Furnari, D. "4 Tips in Data Presentation | Moviri - It's all about performance", *Moviri - It's all about performance*, 2018. [Online]. Available: <https://www.moviri.com/2013/12/top-tips-data-presentation/>. [Accessed: 11- Jan- 2018].
- [19] Pflueger, J., "Understanding Data Center Energy Intensity", *Thegreengrid.org*, 2010. [Online]. Available: <https://www.thegreengrid.org/file/432/download?token=rDHFXQUs>. [Accessed: 11- Jan- 2018]
- [20] Jiang, N, Parashar, M. 2009, March "Enabling autonomic power-aware management of instrumented data centers", In *IEEE International Symposium of Parallel Distributed. Processing*, (pp. 1-8).

- [21] Yoder, A.G. 2010, June. "Energy efficient storage technologies for data centers", In *Workshop Energy-Efficient Design (WEED)*, (pp. 13-18).
- [22] Hatanaka, T., Yajima, R., Horiuchi, T., Wang, S., Zhang, X., Takahashi, M., Sakai, S. and Takeuchi, K., 2009, June. "Ferroelectric (Fe)—NAND flash memory with non-volatile page buffer for data center application enterprise solid-state drives (SSD)", In *IEEE Proceedings of VLSI Circuits Symposium.*, (pp. 78-79).
- [23] Zhou, P., Zhao, B., Yang, J. and Zhang, Y., "A durable and energy efficient main memory using phase change memory technology", In *ACM SIGARCH Computer Architecture. News*, 37(3), 2009, pp. 14-23.
- [24] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y. and Lu, S. "DCell: A scalable and fault-tolerant network structure for data centers", In *Proceedings of ACM SIGCOMM Conference on Data Communication Review.*, 38(4), 2008, pp. 75-86.
- [25] Al-Fares, M., Loukissas, A. and Vahdat, A, "A scalable commodity data center network architecture", In *SIGCOMM Computer Communication Review*. 38(4), 2008, pp. 63-74.
- [26] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y. and Lu, S, "BCube: A high performance server-centric network architecture for modular data centers", In *SIGCOMM Computer Communication Review*. 39(4), 2009, pp. 63-74.
- [27] Lu, G., Guo, C., Li, Y., Zhou, Z., Yuan, T., Wu, H., Xiong, Y., Gao, R. and Zhang, Y. "Server Switch: A programmable and high performance platform for data center networks", In *Proceedings of NSDI*, 11(2-2), 2011, pp. 1-14.
- [28] Mahadevan, P., Banerjee, S., Sharma, P., Shah, A. and Ranganathan, P. "On energy efficiency for enterprise and data center networks", In *IEEE Communication Magazine*, 49(8), 2011, pp. 94-100.
- [29] Abts, D., Marty, M.R., Wells, P.M., Klausler, P. and Liu, H., "Energy proportional datacenter networks", In *ACM SIGARCH Computer. Architecture. News*, 38(3), 2010, pp. 338-347.
- [30] P. Hodge, "Virtualization 101: Understanding how to do more with less", System 2011
- [31] Hu, B., Lei, Z., Lei, Y., Xu, D. and Li, J. 2011, December. "A time-series based preoccupy approach for live migration of virtual machines", In *IEEE Proceedings of 17th International Conference on Parallel and Distributed Systems (ICPADS)*, (pp. 947-952).

- [32] Heller B, Seetharaman S, Mahadevan P, 2010. "Elastic Tree: Saving Energy in Data Center Network." In *USENIX Proceedings of 7th Conference on Networked Systems Design and Implementation (NSDI)*. (pp 249-264).
- [33] Abts D, Marty MR, Wells PM, Klausler P, Liu H. "Energy Proportional Datacenter Networks". In *IEEE Proceedings of the 37th International Symposium Computer Architecture. Saint-Malo, France, 38(3), 2010*, pp 338–347.
- [34] Carrega A, Singh S, Bruschi R, Bolla R. 2012, July. "Traffic Merging for Energy-Efficient Datacenter Networks" In *Proceedings of IEEE International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, Genoa Italy, (pp 1–5).
- [35] Prete L, Farina F, Campanella M, Biancini A. 2012, October "Energy Efficient Minimum Spanning Tree in OpenFlow Networks". In *Proceedings of IEEE European Workshop on Software Define Networking, Darmstadt Germany*, (pp 36-41).
- [36] Kakadia D, Varma V (2012) "Energy Efficient Data Center Networks – A SDN based Approach." Bangalore, IBM Collaborative Academia Research Exchange (I-CARE) IISC, Report No: IIIT/TR/2012/-1
- [37] Energy.gov. (2017). "Energy Efficiency in Data Centers | Department of Energy". [Online] Available at: <https://energy.gov/eere/femp/energy-efficiency-data-centers> [Accessed 8 Feb. 2017].
- [38] Shivashankarappa, A., Basavanna, S. and Dharmalingam, R., 2015, Aug. "Performance Optimization of a Data Center Using Virtualization Technology: A Case Study Utilizing DEMATEL Method", in *Proceedings of IEEE 3rd International Conference on Future Internet of Things and Cloud*, Rome, Italy, (pp. 564-672).
- [39] Ali, E., Susandri and Rahmaddeni, "Optimizing Server Resource by Using Virtualization Technology", *Procedia Computer Science*, 59(1), 2015, pp. 320-325.
- [40] A. Schlichting, "Data Center Energy Efficiency Technologies and Methodologies", P.hD, MITRE, 2017.
- [41] Choudhary, A., Rana, S. and Matahai, K.J., "A Critical Analysis of Energy Efficient Virtual Machine Placement Techniques and its Optimization in a Cloud Computing Environment", *Procedia Computer Science*, 78, 2016, pp. 132-138.
- [42] Tate, J., Beck, P., Clemens, P., Freitas, S., Gatz, J., Girola, M., Gmitter, J., Mueller, H., O'Hanlon, R., Para, V. and Robinson, J. "IBM and Cisco: Together for a World Class Data Center". IBM Redbooks, 2013.

- [43] Liu, J. and Terzis, A. "Sensing data centers for energy efficiency", *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958), 2011, pp. 136-157.
- [44] Zhang, Q., Cheng, L. Boutaba, R., "Cloud computing: State-of-the-art and research challenges", *Journal of Internet Services and Applications*. 1(1), 2010, pp. 7-18.
- [45] Carlinet, Y. and Perrot, N. 2016, September. "Energy-efficient load balancing in a SDN-based Data-Center network", in *Proceedings of IEEE on International Telecommunications Network Strategy and Planning Symposium (Networks)*, CA, USA (pp. 138-143).
- [48] Sverdlík, Y. "Here's How Much Energy All US Data Centers Consume", *Data Center Knowledge*, 2017. [Online]. Available: <http://www.datacenterknowledge.com/archives/2016/06/27/heres-how-much-energy-all-us-data-centers-consume>. [Accessed: 16- Aug- 2017].
- [49] Networking, Cisco Visual. "Cisco Global Cloud Index: forecast and methodology, 2015-2020. White paper." (2017).
- [50] Ghuman, Karanjot Singh. "*Improving Energy Efficiency and Bandwidth Utilization in Data Center Networks Using Segment Routing*". Dissertation. Université d'Ottawa/University of Ottawa, 2017.
- [51] Chen, L., Qiu, M. and Xiong, J., 2015, November. "An SDN-based fabric for flexible data-center networks". In *Proceedings of IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)*, (pp. 121-126).
- [52] Gershman, A. and Sidiropoulos, N. eds., 2005, Aug. "*Space-time processing for MIMO communications*." John Wiley & Sons.
- [53] Bolla, R., Bruschi, R., Davoli, F. and Cucchietti, F. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. In *IEEE Communications Surveys & Tutorials*, 13(2), 2011, pp.223-244.
- [54] Zhang, M., Yi, C., Liu, B. and Zhang, B. 2010, October. "GreenTE: power-aware traffic engineering," in *IEEE proceedings of 18th International Conference on Network Protocols (ICNP)*, (pp. 21-30).
- [55] West, G., "Cisco Spine and Leaf Architecture Discussion - Nexus 5500 vs 6001", *The Network Surgeon*, 2017. [Online]. Available: <http://thenetworksurgeon.com/cisco-spine-and-leaf-architecture-discussion-nexus-5500-vs-6001/>. [Accessed: 02- Sep- 2017].

- [56] Liu, Y., Muppala, J.K., Veeraraghavan, M., Lin, D. and Hamdi, M., “*Data center networks: Topologies, architectures and fault-tolerance characteristics*”. Springer Science & Business Media, 2013.
- [57] Arista Networks, "Software Driven Cloud Networking", *Arista Networks*, 2017. [Online]. Available: <https://www.arista.com/en/>. [Accessed: 03- Sep- 2017].
- [58] *Arista 7500 “Scale-Out Cloud Network Designs”*, 1st ed. Santa Clara, California: Arista.com, 2017, pp. 1-18.
- [59] Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S., "Software-defined networking: A comprehensive survey." In *Proceedings of the IEEE*, 103(1), 2015, pp. 14-76.
- [60] Docs.opendaylight.org. (2017). *Welcome to the OpenDaylight Handbook! — OpenDaylight Documentation Nitrogen documentation*. [online] Available at: <http://docs.opendaylight.org/en/stable-nitrogen/index.html> [Accessed 23 Jun. 2017].
- [61] Medved, J., Varga, R., Tkacik, A. and Gray, K., 2014, June. “Opendaylight: Towards a model-driven SDN controller architecture”. In *IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, (pp. 1-6).
- [62] Lai, W., and F. Le Faucheur. "Requirements for support of differentiated services-aware MPLS traffic engineering." (2003).
- [63] Das, Saurav. *PAC. C: “A unified control architecture for packet and circuit network convergence”*. Dissertation Stanford University, 2012.
- [64] Xia, W., Wen, Y., Foh, C.H., Niyato, D. and Xie, H., "A survey on software-defined networking." *IEEE Communications Surveys & Tutorials* 17(1), 2015, pp. 27-51.
- [65] "Open vSwitch", *Openvswitch.org*, 2017. [Online]. Available: <http://openvswitch.org/>. [Accessed: 04-Oct- 2015].
- [66] Cisco, P. and Cisco E, "Chapter 5 MPLS VPN QoS Design", in “*End-to-End QoS Network Design: quality of Service for Rich-Media & Cloud Networks*”, 3rd ed., C. Systems, Ed. Carlifornia: Cisco Press, 2015, pp. 5-1 to 5-32.
- [67] Kim, H., and Feamster, N. "Improving network management with software defined networking", In *IEEE Communications Magazine*, 51(2), 2013, pp. 114-119.

- [68] "Segment Routing", *Segment Routing*, 2017. [Online]. Available: <http://www.segment-routing.net/>. [Accessed: 07- Sep- 2017].
- [69] Bonica, R. *A Segment Routing (SR) Tutorial*, 70th ed. Juniper Networks, 2017, pp. 1-44.
- [70]"Common Configuration Issues — Open vSwitch 2.8.90 documentation", *Docs.openvswitch.org*, 2017. [Online]. Available: <http://docs.openvswitch.org/en/latest/faq/issues/>. [Accessed: 11- Sep- 2017]
- [71] Team, M., "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet", *Mininet.org*, 2017. [Online]. Available: <http://mininet.org/>. [Accessed: 16- Sep- 2017].
- [72] "Cisco Live sessions, videos and exhibitors from all global conferences", *Ciscolive.com*, 2017. [Online]. Available: <https://www.ciscolive.com/online/connect/search.wv>. [Accessed: 18- Aug- 2017].
- [73] DNA, N. "SD-WAN - Software-Defined WAN and IWAN", *Cisco*, 2017. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/enterprise-networks/intelligent-wan/index.html#~stickynav=1>. [Accessed: 05- Oct- 2017].
- [74] Filsfil, C., Nainar, N.K., Pignataro, C., Cardona, J.C. and Francois, P., 2015, December. "The segment routing architecture". In *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, (pp. 1-6).
- [75] "DiffServ -- The Scalable End-to-End QoS Model & nbsp; [QoS Signaling]", *Cisco*, 2017. [Online]. Available: https://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper09186a00800a3e2f.html. [Accessed: 11- Oct- 2017].
- [76] P. Services, C. Distribution, C. Switches and W. Papers, "Buffers, Queues & Thresholds on Catalyst 6500 Ethernet Modules", *Cisco*, 2017. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/prod_white_paper09186a0080131086.html. [Accessed: 06- Oct- 2017].
- [77] Giorgetti, A., Sgambelluri, A., Paolucci, F., Cugini, F. and Castoldi, P., "Segment routing for effective recovery and multi-domain traffic engineering". *Journal of Optical Communications and Networking*, 9(2), 2017, pp.A223-A232.
- [78] Frost, D.C., Bryant, S.F. and Filsfil, C., Cisco Technology, Inc., 2017. "MPLS segment-routing". U.S. Patent 9,749,227.

- [79] Filtsils, C., Bryant, S.F., Frost, D.C. and Evans, J.W., Cisco Technology, Inc., 2013. “*Segment Routing Techniques*”. U.S. Patent Application 14/047,310.
- [80] Sheu, J.P. and Chen, Y.C., 2017, May. “A scalable and bandwidth-efficient multicast algorithm based on segment routing in software-defined networking”. In *Proceedings of IEEE International Conference on Communications (ICC)*, (pp. 1-6).
- [81] Docs.opendaylight.org, Copyright Opendaylight "Controller Projects' Modules/Bundles and Interfaces OpenDaylightProject", *Wiki.opendaylight.org*, 2017. [Online]. Available: https://wiki.opendaylight.org/view/Controller_Projects%27_Modules/Bundles_and_Interfaces. [Accessed: 05- Oct- 2017].
- [82] S. Cisco, *Segment Routing Configuration Guide*, 1st ed. San Jose California USA. Cisco Systems 2017, pp. 1-200.
- [83]. Yedder, H.B., Ding, Q., Zakia, U., Li, Z., Haeri, S. and Trajkovic, L., 2017, July. “Comparison of Virtualization Algorithms and Topologies for Data Center Networks”. In *proceedings of IEEE, 26th International Conference Computer Communication and Networks (ICCCN)*, (pp. 1-6).
- [84] Wang, T., Su, Z., Xia, Y. and Hamdi, M., 2014. “Rethinking the data center networking: Architecture, network protocols, and resource sharing”. *IEEE access*, 2, pp.1481-1496.
- [85] Hammadi, A. and Mhamdi, L. “A survey on architectures and energy efficiency in data center networks”. *Computer Communications*, 40, 2014, pp.1-21.
- [86] Haeri, S. and Trajković, L., 2016, May. “Virtual network embedding’s in data center networks”. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, (pp. 874-877).
- [87] Chen, L., Qiu, M., Dai, W. and Jiang, N., “Supporting high-quality video streaming with SDN-based CDNs”. *The Journal of Supercomputing*, 73(8), 2017, pp.3547-3561.
- [88] Biswas, A., Hussain, M.A. and Kalita, A., 2016, October. “An improved congestion free modified fat tree Network on—Chip Topology”. In *Proceedings of IEEE International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, (pp. 759-763).
- [89] Zafar, S., Bashir, A. and Chaudhry, S.A. “On implementation of DCTCP on three-tier and Fat-tree data center network topologies”. *SpringerPlus*, 5(1), 2016, p.766.
- [90] Support, T., Routing, I. and TechNotes, T. (2017). *How Does Unequal Cost Path Load Balancing (Variance) Work in IGRP and EIGRP?*. [online] Cisco. Available at:

<https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13677-19.html> [Accessed 15 Dec. 2017].

[91] C. Services, D. Switches, C. Switches and W. Papers, "Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper", *Cisco*, 2017. [Online]. Available: <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html>. [Accessed: 04- May- 2017].

[92] Sun, Y., Chen, J., Lu, Q. and Fang, W. "Diamond: an improved Fat-tree architecture for large-scale data centers". *Journal of Communications*, 9(1), 2014, pp.91-98.

[93] Wang, B., Qi, Z., Ma, R., Guan, H. and Vasilakos, A.V., 2015. "A survey on data center networking for cloud computing". *Computer Networks*, 91, 2015, pp.528-547.

[94] A. Andreyev. (2014, Nov.) Introducing data center fabric, the next-generation Facebook data center network. [Online]. Available: <https://code.facebook.com/posts/360346274145943>. [Accessed: 04-Nov-2017].

[95] Chen, G., Zhao, Y., Xu, H., Pei, D. and Li, D, "F2Tree: Rapid Failure Recovery for routing in Production Data Center Networks", in *IEEE Transactions on Networking*, 25(4), 2017, pp. 1940 - 1953.

[96] Farrington, N. and Andreyev, A., 2013, May. "Facebook's data center network architecture". In *Proceedings of IEEE on Optical Interconnects Conference, 2013 IEEE* (pp. 49-50).

[97] Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J. and Wright, N.J., 2010, November. "Performance analysis of high performance computing applications on the amazon web services cloud". In *Proceedings of IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, (pp. 159-168).

[98] Grossman, R.L., "The case for cloud computing". *IT professional*, 11(2), 2009, pp.23-27.

[99] Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y., Srivastav, S., Wu, J., Simitci, H. and Haridas, J., 2011, October. Windows Azure Storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (pp. 143-157).

- [100] Rahman, S., Gupta, A., Tomatore, M. and Mukherjee, B., 2017, May. "Dynamic workload migration over optical backbone network to minimize data center electricity cost". In *Proceedings of IEEE International Conference on Communications (ICC)*, (pp. 1-5).
- [101] Kim, I., Kim, T. and Eom, Y.I., 2010, March. NHVM: design and implementation of Linux server virtual machine using hybrid virtualization technology. In *Proceedings of IEEE International Conference on Computational Science and Its Applications (ICCSA), 2010* (pp. 171-175).
- [102] Docs.opendaylight.org. (2017). *Link Aggregation Control Protocol User Guide — OpenDaylight Documentation Nitrogen documentation*. [Online] Available at: <http://docs.opendaylight.org/en/stable-nitrogen/user-guide/link-aggregation-control-protocol-user-guide.html> [Last Accessed 11 Dec. 2017].
- [103] Li, D., Wu, J., Liu, Z. and Zhang, F., 2015, September. "Dual-centric data center network architectures". In *Proceedings of IEEE 44th International Conference on Parallel Processing (ICPP)*, (pp. 679-688).
- [104] Feamster, N., Rexford, J. and Zegura, E. The road to SDN. *Queue*, 11(12), 2013, p.20.

Appendices A

Images

ODL view (YANG API) of Multiple Leaf-Spine Network (4-10-1 spine-leaf-hosts)

ODL view (YANG) of 4-4-4 spines-leaves- and hosts

3)

entire network 8 spines 96 leafs fan-out 1 as seen by controller

5) Some APIs⁸ used

odl-ndsal-don	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: DOM
odl-ndsal-common	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: DOM Broker
odl-ndsal-don-api	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: DOM API & SPI
odl-ndsal-don-broker	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: DOM Broker
odl-ndsal-binding-base	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Binding Base Concepts
odl-ndsal-binding-runtime	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Binding Generator
odl-ndsal-binding-api	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Binding Base Concepts
odl-ndsal-binding-don-adapter	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Binding Generator
odl-ndsal-eos-common	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: EOS :: Common
odl-ndsal-eos-don	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: EOS :: DOM
odl-ndsal-eos-binding	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: EOS :: Binding
odl-ndsal-singleton-common	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Singleton :: Common
odl-ndsal-singleton-don	2.2.1-Carbon	x	odl-yangtools-2.2.1-Carbon	OpenDaylight :: MD-SAL :: Singleton :: DOM
odl-ovsdb-southbound-api	1.4.1-Carbon	x	odl-ovsdb-southbound-1.4.1-Carbon	OpenDaylight :: southbound :: api
odl-bgpcep-bmp	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-restconf-all	1.5.1-Carbon	x	odl-controller-1.5.1-Carbon	OpenDaylight :: Restconf :: All
odl-restconf	1.5.1-Carbon	x	odl-controller-1.5.1-Carbon	OpenDaylight :: Restconf
odl-restconf-noauth	1.5.1-Carbon	x	odl-controller-1.5.1-Carbon	OpenDaylight :: Restconf
odl-ndsal-apidocs	1.5.1-Carbon	x	odl-controller-1.5.1-Carbon	OpenDaylight :: MDSAL :: APIDOCs
odl-nic-core-ndsal	1.3.1-Carbon	x	odl-nic-1.3.1-Carbon	OpenDaylight :: Network Intent Composition :: Core
odl-nic-renderer-vtn	1.3.1-Carbon	x	odl-nic-1.3.1-Carbon	OpenDaylight :: Network Intent Composition :: VTN
odl-nic-renderer-of	1.3.1-Carbon	x	odl-nic-1.3.1-Carbon	OpenDaylight :: Network Intent Composition :: Open
odl-nic-pipeline-manager	1.3.1-Carbon	x	odl-nic-1.3.1-Carbon	OpenDaylight :: Network Intent Composition :: Open
pax-jetty	8.1.19.v20160209	x	org.ops4j.pax.web-3.2.9	Provide Jetty engine support
pax-http	3.2.9	x	org.ops4j.pax.web-3.2.9	Implementation of the OSGi HTTP Service
pax-http-whiteboard	3.2.9	x	org.ops4j.pax.web-3.2.9	Provide HTTP Whiteboard pattern support
pax-war	3.2.9	x	org.ops4j.pax.web-3.2.9	Provide support of a full WebContainer
odl-netty	4.1.1.Final	x	odiparent-1.8.1-Carbon	OpenDaylight :: Netty
odl-netty-4	1.8.1-Carbon	x	odiparent-1.8.1-Carbon	OpenDaylight :: Netty
odl-guava-18	1.8.1-Carbon	x	odiparent-1.8.1-Carbon	OpenDaylight :: Guava 18
odl-lmax-3	1.8.1-Carbon	x	odiparent-1.8.1-Carbon	OpenDaylight :: LMAX Disruptor
odl-triemap-0.2	1.8.1-Carbon	x	odiparent-1.8.1-Carbon	OpenDaylight :: Concurrent Trie HashMap
odl-ndsal-common	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	OpenDaylight :: Config :: All
odl-ndsal-broker-local	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	OpenDaylight :: MDSAL :: Broker
odl-ndsal-clustering-commons	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	
odl-ndsal-distributed-datastore	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	
odl-ndsal-remoteipc-connector	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	
odl-ndsal-broker	1.5.1-Carbon	x	odl-ndsal-1.5.1-Carbon	
odl-aaa-shiro	0.5.1-Carbon	x	odl-aaa-0.5.1-Carbon	OpenDaylight :: AAA :: Shiro
odl-bgpcep-bgp	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-config-files	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-openconfig	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-dependencies	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-inet	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-parser	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-rib-api	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-linkstate	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-flowspec	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-labeled-unicast	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-l3vpn	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-evpn	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-path-selection-node	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-rib-impl	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-topology	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-cli	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-config-loader	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-bgpcep-bgp-openconfig-state	0.7.1-Carbon	x	odl-bgpcep-0.7.1-Carbon	
odl-l2switch-all	0.5.1-Carbon	x	l2switch-0.5.1-Carbon	OpenDaylight :: L2Switch :: All
odl-l2switch-switch	0.5.1-Carbon	x	l2switch-0.5.1-Carbon	OpenDaylight :: L2Switch :: Switch
odl-l2switch-switch-rest	0.5.1-Carbon	x	l2switch-0.5.1-Carbon	OpenDaylight :: L2Switch :: Switch

2. segment routing auto-bandwidth in action

While running the network and transmitting simultaneously same UDP packet across the network, we see the segment routing routing algorithm in action on leaf switch 1, excerpt take from Linux Network properties

110

3. Routing packets across a leaf switch


```

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:4068 errors:0 dropped:22 overruns:0 frame:0
TX packets:111242 errors:0 dropped:6347 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:283980 (283.9 KB) TX bytes:7746148 (7.7 MB)

s1-eth20 Link encap:Ethernet HWaddr ae:15:bc:4d:9b:4b
        inet6 addr: fe80::ac15:bcff:fe4d:9b4b/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4082 errors:0 dropped:24 overruns:0 frame:0
        TX packets:111257 errors:0 dropped:6362 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:284960 (284.9 KB) TX bytes:7748206 (7.7 MB)

s1-eth21 Link encap:Ethernet HWaddr ca:df:24:8d:76:d4
        inet6 addr: fe80::c8df:24ff:fe8d:76d4/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4067 errors:0 dropped:26 overruns:0 frame:0
        TX packets:111250 errors:0 dropped:6356 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:284694 (284.6 KB) TX bytes:7746820 (7.7 MB)

s1-eth22 Link encap:Ethernet HWaddr c6:9f:f2:ed:bb:28
        inet6 addr: fe80::c49f:f2ff:feed:bb28/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4058 errors:0 dropped:26 overruns:0 frame:0
        TX packets:111286 errors:0 dropped:6351 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:283896 (283.8 KB) TX bytes:7749788 (7.7 MB)

s1-eth23 Link encap:Ethernet HWaddr ca:9e:3d:18:51:1f
        inet6 addr: fe80::c89e:3dff:fe18:511f/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4067 errors:0 dropped:26 overruns:0 frame:0
        TX packets:111277 errors:0 dropped:6344 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:285030 (285.0 KB) TX bytes:7748430 (7.7 MB)

s1-eth24 Link encap:Ethernet HWaddr 92:74:8e:18:df:af
        inet6 addr: fe80::9074:8eff:fe18:dfaf/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4060 errors:0 dropped:32 overruns:0 frame:0
        TX packets:111311 errors:0 dropped:6337 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:284596 (284.5 KB) TX bytes:7751538 (7.7 MB)

s1-eth25 Link encap:Ethernet HWaddr 9a:f1:74:6a:4d:80
        inet6 addr: fe80::98f1:74ff:fe6a:4d80/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4044 errors:0 dropped:26 overruns:0 frame:0
        TX packets:111313 errors:0 dropped:6333 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:282580 (282.5 KB) TX bytes:7751734 (7.7 MB)

s1-eth26 Link encap:Ethernet HWaddr 2a:ef:42:d3:5e:e3
        inet6 addr: fe80::28ef:42ff:fed3:5ee3/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:4046 errors:0 dropped:26 overruns:0 frame:0
        TX packets:111287 errors:0 dropped:6330 overruns:0 carrier:0

```

system performance picture when transmitting simultaneously same UDP packet across network with segment routing auto-bandwidth in action.

4.

system performance without segment routing in action

Zoomed view

```

Node: h1
UDP buffer size: 208 KByte (default)
[247] local 10.0.0.1 port 57572 connected with 10.0.23.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[247] 0.0-180.0 sec  858 MBytes   40.0 Mbits/sec
[247] Sent 612246 datagrams
[247] Server Report:
[247] 0.0-180.0 sec  858 MBytes  40.0 Mbits/sec  0.001 ms  0/612245 (0%)
[247] 0.0-180.0 sec  1 datagrams received out-of-order
root@lung-07-l:~# iperf -c 10.0.23.2 -u -b 150m -t 180

Client connecting to 10.0.23.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[247] local 10.0.0.1 port 56630 connected with 10.0.23.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[247] 0.0-180.0 sec  2.03 GBytes  97.1 Mbits/sec
[247] Sent 1485702 datagrams
[247] Server Report:
[247] 0.0-180.0 sec  2.03 GBytes  97.1 Mbits/sec  0.373 ms  32/1485701 (0.002
2%)
[247] 0.0-180.0 sec  148 datagrams received out-of-order
root@lung-07-l:~#

[247] local 10.0.19.1 port 51407 connected with 10.0.5.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[247] 0.0-180.0 sec  858 MBytes   40.0 Mbits/sec
[247] Sent 612246 datagrams

Node: h19
UDP buffer size: 208 KByte (default)
[247] local 10.0.9.1 port 47572 connected with 10.0.17.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[247] 0.0-180.0 sec  858 MBytes   40.0 Mbits/sec
[247] Sent 612246 datagrams
[247] Server Report:
[247] 0.0-180.0 sec  858 MBytes  40.0 Mbits/sec  0.000 ms  0/612245 (0%)
[247] 0.0-180.0 sec  1 datagrams received out-of-order
root@lung-07-l:~# iperf -c 10.0.17.2 -u -b 150m -t 180

Client connecting to 10.0.17.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[247] local 10.0.9.1 port 49204 connected with 10.0.17.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[247] 0.0-180.0 sec  2.03 GBytes  97.1 Mbits/sec
[247] Sent 1485511 datagrams
[247] Server Report:
[247] 0.0-180.0 sec  2.03 GBytes  97.0 Mbits/sec  0.452 ms  98/1485510 (0.006
6%)
[247] 0.0-180.0 sec  99 datagrams received out-of-order
root@lung-07-l:~#

Node: h6
[247] local 10.0.2.2 port 47316 connected with 10.0.20.2 port 5001

```

And with no SR on

Further images no SR only Dijkstra iFwd API

1)

```
"Node: h1"
root@lung-07-l:~# iperf -c 10.0.3.4 -u -b 150m -t 180
-----
Client connecting to 10.0.3.4, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 55] local 10.0.0.1 port 60294 connected with 10.0.3.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 55] 0.0-180.0 sec  2.05 GBytes  98.0 Mbits/sec
[ 55] Sent 1499694 datagrams
[ 55] Server Report:
[ 55] 0.0-180.2 sec  499 MBytes  23.2 Mbits/sec   0.412 ms 1144012/1499693 (76%)
[ 55] 0.0-180.2 sec  24 datagrams received out-of-order
root@lung-07-l:~#

"Node: h16"
[ 55] 120.0-125.0 sec  2 datagrams received out-of-order
[ 55] 125.0-130.0 sec  25.2 MBytes  42.2 Mbits/sec   0.219 ms 23395/41351 (57%)
[ 55] 130.0-135.0 sec  17.4 MBytes  29.2 Mbits/sec   1.097 ms 28848/41277 (70%)
[ 55] 130.0-135.0 sec  1 datagrams received out-of-order
[ 55] 135.0-140.0 sec  27.4 MBytes  46.0 Mbits/sec   0.111 ms 21834/41373 (53%)
[ 55] 135.0-140.0 sec  1 datagrams received out-of-order
[ 55] 140.0-145.0 sec  21.6 MBytes  36.3 Mbits/sec   0.466 ms 25852/41282 (63%)
[ 55] 145.0-150.0 sec  4.95 MBytes  8.31 Mbits/sec   1.377 ms 37754/41287 (91%)
[ 55] 145.0-150.0 sec  1 datagrams received out-of-order
[ 55] 150.0-155.0 sec  3.91 MBytes  6.55 Mbits/sec   1.693 ms 39495/41281 (93%)
[ 55] 155.0-160.0 sec  14.1 MBytes  23.7 Mbits/sec   0.483 ms 31284/41349 (76%)
[ 55] 155.0-160.0 sec  1 datagrams received out-of-order
[ 55] 160.0-165.0 sec  7.86 MBytes  13.2 Mbits/sec   0.528 ms 35646/41256 (86%)
[ 55] 160.0-165.0 sec  1 datagrams received out-of-order
[ 55] 165.0-170.0 sec  14.4 MBytes  24.2 Mbits/sec   1.193 ms 31025/41294 (75%)
[ 55] 165.0-170.0 sec  1 datagrams received out-of-order
[ 55] 170.0-175.0 sec  3.45 MBytes  5.78 Mbits/sec   0.175 ms 38901/41393 (94%)
[ 55] 175.0-180.0 sec  25.9 MBytes  43.5 Mbits/sec   0.353 ms 22818/41308 (55%)
[ 55] 175.0-180.0 sec  1 datagrams received out-of-order
[ 55] 0.0-180.2 sec  499 MBytes  23.2 Mbits/sec   0.412 ms 1144012/1499693 (76%)
[ 55] 0.0-180.2 sec  24 datagrams received out-of-order
read failed: Connection refused
^
```


2)

3)