Improving Energy Efficiency and Bandwidth Utilization in Data Center Networks Using Segment Routing

by

Karanjot Singh Ghuman

A Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

in Electrical and Computer Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
University of Ottawa
Ottawa, Canada

December 2016

Abstract

In today's scenario, energy efficiency has become one of the most crucial issues for Data Center Networks (DCN). This paper analyses the energy saving capability of a Data center network using Segment Routing (SR) based model within a Software Defined Network (SDN) architecture. Energy efficiency is measured in terms of number of links turned off and for how long the links remain in sleep mode. Apart from saving the energy by turning off links, our work further efficiently manages the traffic within the available links by using Per-packet based load balancing approach. Aiming to avoid congestion within DCN's and increase the sleeping time of inactive links. An algorithm for deciding the particular set of links to be turned off within a network is presented. With the introduction of per-packet approach within SR/SDN model, we have successfully saved 21 % of energy within DCN topology. Results show that the proposed Per-packet SR model using Random Packet Spraying (RPS) saves more energy and provides better performance as compared to Per-flow based SR model, which uses Equal Cost Multiple Path (ECMP) for load balancing. But, certain problems also come into picture using per-packet approach, like out of order packets and longer end to end delay. To further solidify the effect of SR in saving energy within DCN and avoid previously introduced problems, we have used per-flow based Flow Reservation approach along with a proposed Flow Scheduling Algorithm. Flow rate of all incoming flows can be deduced using Flow reservation approach, which is further used by Flow Scheduling Algorithm to increase Bandwidth utilization Ratio of links. Ultimately, managing the traffic more efficiently and increasing the sleeping time of links, leading to more energy savings. Results show that, the energy savings are almost similar in per-packet based approach and per-flow based approach with bandwidth reservation. Except, the average sleeping time of links in per-flow based approach with bandwidth reservation decreases less severely as compared to per-packet based approach, as overall traffic load increases.

Acknowledgements

This thesis has been a challenging and wonderful journey. I take this opportunity to thank Faculty of Graduate and Postdoctoral Studies, University Of Ottawa for giving me this opportunity.

Such an endeavour would not have been possible without the much appreciated help and support of the people surrounding me. To begin, I owe a debt of gratitude to my thesis supervisor, Dr. Amiya Nayak. His guidance and expertise has been invaluable, while demanding the most out of me academically and spurring my personal growth. I consider myself lucky to have received his esteemed guidance as well as his empathy in regard to every situation throughout my research period.

I'd like to acknowledge Radu Carpa for his valuable suggestions and guidance. His expertise in the field of my research topic, have greatly influenced my work. Further, I thank Armaan Sekhon, Amardev Khokhar and Aman Hunjan for their support and guidance throughout my studies.

An individual is not without the heavy influence of their family. Mom – thanks for your truly unconditional love. Dad – giving me my analytic mind and dedicated personality that got me here… and let me finish. Amrit and Jeevan – the best sisters I could ask for. Thanks for your love and support, throughout my life. I owe a wealth of thanks to Yashika Panjeta, for emotional support and technical guidance provided by her.

To my immediate colleagues – Sarabjeet Singh, Mohit Sain and Pranay Sharma– I wish you all the best in your continuing pursuits and thank each of you for helping to make me a better student and person.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ALR	Adaptive Link Rate
APIs	Application Programming Interfaces
AQM	Active Queue Management
B	Bytes
BGP	Border Gateway Protocol
CO_2	Carbon dioxide
CPU	Central Processing Unit
CR-LDP	Constraint based Label Distribution Protocol
CSPF	Constrained Shortest Path First
DCN	Data Center Network
DCTCP	Datacenter Transmission Control Protocol
ECMP	Equal Cost Multi Path protocol
EDF	Earliest Deadline First
EXR	Exclusive Routing
FCT	Flow Completion Time
FEC	Forwarding Equivalence Class
FIB	Forwarding Information Base
FRR	Fast Re-Route
GB	Giga Byte
Gbps	Giga Byte per second
GHz	Giga Hertz
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IEP	Ingress Egress Pair
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
INET	Internetworking
IP	Internet Protocol
IPV6	Internet Protocol version 6
ISP	Internet Service Provider
IS-IS	Intermediate System to Intermediate System

IT	Information Technology
KB	Kilo Byte
LDP	Label Distribution Protocol
LSA	Link-State Advertisement
LSP	Link State Packet
MB	Mega Byte
Mbps	Mega Byte per second
MPLS	Multi-Protocol Label Switching
MP-TCP	Multi Path Transmission Control Protocol
ms	Millisecond
NIC	Network Interface Card
NTC	Network Traffic Consolidation
NP-Hard	Non-Deterministic Polynomial-time Hard
ONOS	Open Network Operating System
OS	Operating System
OSPF	Open Shortest Path First
PB	Peta Byte
PC	Personal Computer
PCE	Path Computation Element
PHP	Penultimate Hop Popping
PLR	Point of Local Repair
QoS	Quality of Service
RAM	Random Access Memory
RFC	Request For Comment
RPS	Random Packet Spraying
RSVP	Resource Reservation Protocol
RTT	Round Trip Time
SDN	Software Defined Networking
SID	Segment Identifier
SLA	Service Level Agreement
SLC	Server Load Consolidation
SPF	Shortest Path First
SPRING	Source Packet Routing in Networking
SR	Segment Routing

SRGB	Segment Routing Global Block
SRDB	Segment Routing Database
STREETE	Segment Routing based Energy Efficient Traffic Engineering
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TE	Traffic Engineering
TWh/Year	Terawatt Hours per Year
UDP	User Datagram Protocol
VM	Virtual Machine
VOIP	Voice over Internet Protocol
WAN	Wide Area Network
μS	Micro Second

Chapter 1 – Introduction and Motivation

1.1. Introduction

Energy efficient networking has gained increasing attention in the past few years because of more and more emphasis being given on economic and environmental factors, making green networking a key concept in development and research of networking community. Studies have shown a continuously growing demand for energy as well as high performance electrical equipment's within datacenter networks [1], although the network devices are designed to endure peak load leaving most of the equipment under-utilized, with a great chances of introducing energy saving models to utilize the resources within a network in an energy-efficient manner.

With the introduction of SDN paradigm [2], emphasis on a centralized control of the network with a separate control and data plane is proving beneficiary in various aspects of networking. With SDN, applications can treat the network as a logical entity which makes network operators gain unprecedented programmability, automation and network control. Also, the concept of SDN architecture has become of great importance in introducing green networking approaches especially within Internet Service Providers (ISPs) and datacenter networks with large scalability. The Source Packet Routing in Networking (SPRING) or Segment Routing (SR) [3], used in our research work was also developed with its implementation oriented towards working in an SDN architecture, leveraging the advantages provided by the centralized controller and making the concept of source routing feasible in large networks which was not used much in its initial development stages due to scalability issues. Our research work basically constitutes two energy efficiency approaches and in both the approaches, the SR model within an SDN architecture is being used to save energy within data center networks keeping in mind energy as well as bandwidth efficiency. As concluded by Mahadevan et al. [4], energy consumed by a switch and router being dependent on the dynamic as well as fixed part of a switch or router, it is feasible to turn off unused links instead of turning off the whole switch or router avoiding long transition periods. We have also introduced an algorithm for efficiently turning off links within the network keeping in mind various connectivity constraints.

1.2. Motivation

With the increasing impact of internet services over our lives and number of users increasing day by day, the demand for consistently good quality of service amidst of increasing demand and resources is a key issue to be maintained among service providers. According to an estimation done in [5] over a period of 2010-2020, the global wireline Internet traffic will increase by a factor of 16, to approach 250 Exabyte per month. Moreover, the global mobile Internet traffic will grow even faster, approximately 150 times to reach 40 Exabyte per month. Whereas, Internet traffic will grow 3.0-fold from 2015 to 2020, with a compound annual growth rate of 25% [6]. Table 1 shows in details traffic projections of different kinds of networks in the Mature Market (consisting of Japan, North America and Western Europe).

Table 1: Mature market traffic projection (PB/ month) [5]

Year	Mobile Access	Wireline Access	Core network
2010	161	7,727	10,707
2015	3,858	33,879	45,402
2020	14,266	74,462	103,085
2020/2010	89x	9.6x	9.6x

ISPs do have to keep pace with multiple folding traffic demands with time, which also leads to increase in demand of bandwidth as well as storage capacities of switches and routers with increased scalability, ultimately leading to increase in CO_2 emission. To support the efforts of introducing green networking methods, we have oriented our research work to save energy within data center networks using the SDN paradigm along with source routing based Segment Routing approach. We have focused not only on saving energy by turning off links but also by managing the traffic within available links in an efficient manner using per-packet load balancing and flow scheduling approaches. Per-packet load balancing made feasible in a scalable network due to SDN architecture, is used to efficiently solve the problem of balancing the long flows and short flows within a network by improving FCT of short flows leading to more sleeping time for unused links. Flow scheduling helps in managing the flows within network efficiently according to bandwidth, increasing bandwidth utilization of links as well as increasing the sleeping time of turned off links.

1.3. Objectives and Contributions

Following are the major objectives of our research:

The main goal of our research work is to save energy particularly in data center networks and propose a green networking method using the latest technology paradigms like SDN and SR, opening up whole new ways and models to make a DCN more energy efficient as well bandwidth efficient. Based on our targets we can deduce following objectives regarding our research:

- Analyze the effects and outcome of introducing the concept of SDN along with source routing based SR technique in saving energy within DCN.
- Examine the energy efficiency pattern, by not only turning off as many links as possible but implementing techniques to manage the traffic load within available links and avoiding congestion within DCN.
- Analyze the literature regarding SDN and SR to leverage as many advantages as possible from these models in regard to increasing energy savings as well as bandwidth utilization of links.
- Analyze and compare the results achieved by different approaches proposed in our research work.

Following are the major contributions of our research work:

- Analyzed the energy efficiency pattern of a DCN using SDN architecture along with per-flow based SR-model and proposed a link selection algorithm to efficiently turn off links within network keeping in mind connectivity constraint.
- Proposed an SR-model based upon per-packet load balancing approach via SDN to efficiently manage the traffic load within available links and compare the results with the per-flow based SR-model.
- To avoid the shortcomings within per-packet based SR-model, proposed an SR-model based upon flow reservation method and put forth a flow scheduling algorithm, not only to save energy but also to increase the bandwidth utilization ratio of links. Also, analyzed and compared the results of proposed approach with per-packet based SR-

model. Demonstrated the flexibility of approach by implementing it on B-Cube topology as well.

- Demonstrated the achievement of an average 43% of links turned off within a DCN using per-packet based SR-model approach. Achieved almost similar set of results with better bandwidth utilization and longer sleeping time for links using bandwidth reservation approach in per-flow based SR-model.

1.4. Organization of Thesis

Rest of the thesis is organized as follow: Chapter 2 provides a background about the technologies used in the work i.e. software defined networking, energy efficiency, segment routing and data center networking. Chapter 3 presents per-packet based SR-model proposal via SDN, for achieving better FCT of short flows, higher energy efficiency and longer sleeping time for unused links. Chapter 4 describes a per-flow based SR model with Bandwidth Reservation approach via SDN aimed to overcome shortcomings of Chapter 3 approach along with higher Bandwidth utilization Ratio. Chapter 5 concludes the work by giving a summary of the research and future research directions based on the thesis.

Chapter 2 - Background and Related Work

In this chapter, we will be discussing the significance of energy efficiency within networks and different energy efficiency techniques used in our research. Also, we will be giving details about Segment Routing approach around which our research work is oriented, along with the concepts of Software Defined Networking, Per-Packet Load Balancing, and Data Center Networking.

2.1. Energy Efficiency

With the advent of the Information and Communication Technology (ICT), more and more services and applications are getting dependent upon the ICT, penetrating further into people's lives. With this significant boom in the Information technology (IT) and Networking industry, there is a growing demand for electricity to maintain a reliable and meaningful quality of service. Due to unexpected environmental changes like global warming and depletion of traditional energy resources, the energy consumed by ICT industry has become a topic of discussion and improvement in different research groups. More and more emphasis is being given upon saving energy and developing as well as implementing Greener Industrial alternatives. The concept of green networking is being put into more attention due to the growth of customer population, spreading broadband access and increasing number of services being provided by ISPs [1], with wired networks and service infrastructures having tremendous opportunities for improvement. The research efforts for greener networking is not only motivated by environmental factors but also by economic factors, as companies are focusing on reduction in operational costs while maintaining high availability and desired service levels.

2.1.1. Need for Green Information Technology

As more and more attention is being given to different Energy efficiency or Green innovation techniques within the networking and IT field, emphasizing on reducing Energy consumption and operational costs within Data Centers, Electronic Networking equipment and Computers. Data center being the core within different Computing and Networking operations include a

significant number of electronic equipment and computers. A typical data center produces 170 million metric tons of CO_2 worldwide per year as estimated in 2009. The expected emission from data center is projected to reach 670 million metric tons of CO_2 worldwide annually by 2020 [7]. Also, with the advent of Cloud Computing, the need for data centers has significantly increased. Due to more equipment and requirement of high availability, the data centers consume 1.5% of world's energy consumption as estimated in 2012 [8]. Even this much usage of energy within the data centers does not return significant favors as most of the servers remain under-utilized.

Laszewskiet et al. [7] present particular SLA impact factors essential for the GreenIT effort. Although Hardware and Environmental considerations are dependent on the kind of hardware we are using in our data center and on the environment, this equipment being operated in, leads to a certain level of energy consumption and heat production, which directly impacts the environment. Whereas Software factor is something which is directly related to our research work, provides optimization mechanisms for reducing the impact of hardware on the environment during runtime. For example, running certain algorithms in the network leads to lesser energy consumption and more optimized bandwidth usage.

2.1.2. Energy Efficiency in Data Center Networks

Apart from Performance being the key aspect of the Data Center, due to the increasing size and dependence on a large number of services in the data centers, energy efficiency has become a key feature in testing the performance of a Data Center. Several energy-aware approaches and resource management techniques have been introduced to tackle power consumption problem in the data center domain, trying to optimize the energy usage from different points of view.

Meisner et al. [9] present an approach to reduce energy consumption, where the network components can make transitions between high-performance active state and a near-zero power idle state depending on the instantaneous load. This approach is less complex because of the emphasis laid on simpler optimization goals, which are minimizing the idle energy and transition time, than the complex load- balancing operation for each system component. Carrol et al. [10], whereas emphasize on saving energy by minimizing the energy consumed by

cooling systems, putting efforts in finding solutions for optimal workload placement. This approach uses Generic Algorithm to move data between data centers located in different geographical locations with a different environment, aiming to move services to an optimal location regarding minimized cooling costs, renewable energy consumption, and service requests.

A lot of other research works have been done in the field of using efficient placement algorithms to reduce the energy consumption or to move the applications to those Data Centers located in areas where the cost of energy is relatively low. But in our research work, we will concentrate on using an energy efficient algorithm to minimize the traffic onto certain links and routers. We can divide Energy efficient data center algorithms into two classes [11]:

(i) Virtual Machine (VM) placement algorithms for consolidating Virtual Machines onto the fewest number of servers
(ii) Efficient network routing algorithms that attempt to do energy efficient routing by consolidating the network traffic onto the smallest number of links.

Meng et al. [12] present an algorithm solving the problem for VM placement within a Data Center network of already known traffic matrix and the cost of communication between any pair of servers. The algorithm uses two-tier heuristic approach to solve the NP-hard problem of VM placement, by partitioning hosts and VM's into clusters separately and then matching VM and hosts at cluster levels and individual levels such that the total communication cost reduces. This approach is dependent upon the type of network architecture is used, makes it very much scalable, but it does not consider network power optimization. Whereas, McGeer et al. [13] not only use the technique of VM placement but simultaneously turns off as many switches as possible to save network power. The VM placement problem is solvable by linear-time heuristic where the switches are turned off at first and then site virtual machine in data center turning on as few switches as possible keeping in mind connectivity constraint and desired bandwidth. However, this approach is designed specifically for the networks with a Fat-Tree topology.

Within our Research work, we have particularly focused on developing an Energy efficient routing algorithm attempting to turn off as many links as possible, alongside managing the traffic within the available links as long as possible. A lot of Energy efficient

routing algorithms have been developed. Nedevschi et al. [14] proposed an algorithm based on buffer-and-burst scheme, called practB&B. The Burst method is used to increase the inter-packet arrival time so that the switches can be turned off for a significant amount of time but this improvement is practically impossible below 100Gbps links and comes at the cost of an added end-to-end delay. Vasic et al. [15] use Energy-aware Traffic Engineering approach to achieve similar load among multiple paths for energy saving using an online technique. Load balancing is achieved by leveraging Rate adaptation, resulting in the uniform distribution of operating rates saves more energy as compared to an exponential distribution and achieving 21% of links as well as 16% of routers to be put into sleep mode.

As discussed above and in other related research studies, some algorithms are focusing on turning off underutilized network elements including the components to provide redundancy within the network by routing the traffic on a particular set of links and routers and other are focusing on limiting the link data rates according to the load being routed on them. Based on this we can divide reduction of energy consumption into two approaches [16]:

First one is to limit the traffic to the specific set of links and powering down the network elements in unused paths. Mahadevan et al. [17] use this approach for energy optimization for a single administrative domain network, concluding that power consumption is mainly determined by active ports and bandwidth capacity configured for each port and uses Network traffic consolidation (NTC) & Server load consolidation (SLC) scheme. NTC is a traffic engineering approach that merges all traffic to fewer links and switches while the unused ones can be deactivated and SLC consolidates loads onto smaller amount of servers to make less links and switches used.

Second one is adapting link rates according to the offered traffic loads reducing the energy consumption by links and ports. Gunaratne et al. [18] suggest lowering the link rates during low utilization periods will result in reducing the energy consumption in Network Interface Cards (NICs). Also, an energy savings of around 0.93 TWh/year is possible by operating 100 million Adaptive Link Rate (ALR) capable Ethernet links. Heller et al. [19] use its ElasticTree approach in both the above cases and concludes that more energy can be saved by consolidating traffic on specific links and turning off unused components. This network-wide energy optimizer scheme results in 25-40% of the network energy savings but also

suggests turning off network components will lead to further cost reduction in cooling processes. Tucker et al. [20] also show that 54% of energy is consumed by the Data plane, out of which 32% is used up for IP look-ups and forwarding engines. Thus, it concludes that turning off network components or links and consolidating traffic on available links saves more energy as compared to adapting link data rates.

Thus, as concluded above from this section we have particularly focused on developing an Energy efficient routing algorithm, to route the traffic onto particular set of available links. Focussing on keeping the occupation ratio of active links to be high and turning off the unused links instead of adaptively changing the links rates, which will save a significant amount of energy.

2.1.3. Power Consumption Model

In the above section, we have analyzed different energy efficiency schemes and built our basis for the Energy efficiency approach; we are going to use in our research work. Within this section, we will discuss significant power consuming sections within a network and on particularly what sections we will be focusing to turn off in our research work to accurately quantify the savings from our energy saving approach. Mahadevan et al. [4] conclude that energy consumed by a switch and router is not proportional to the traffic load or the packet size but is directly dependent on the dynamic as well as fixed part of a switch or router. Taking the example of Cisco Nexus 2224TP [21] switch and Cisco 7507 router [4], Nexus 2224TP switch consists of 1 line card and 24 Gigabit port where fixed parts like chassis, fans, etc. consumes 48W of power, and each port consumes 2W individually counting up to 48W by all the ports combined. Whereas, 7507 routers fixed part consumes 210 watts and depending on the line card port is used on the router power consumption vary from 26W for 1-port Fast Ethernet card to 30W for 1-port Gigabit Ethernet card.

Recommended by IEEE 802.3az[22], turning off fixed parts like fans, chassis, switching fabric, etc. is not very much feasible at frequent rates within large networks due to significant delays produced while turning a switch or router on or off, subsequently affecting the quality of service. As power consumption also depends upon the number of active ports as well as the

line speed of each port, if we can put the inactive or underutilized ports into sleep mode we can save a significant amount of energy within a network.

Total power consumed within a network can be calculated using following equation:

$$E = \sum_{s \in Z} (F_s \times T_s + \sum_{p \in P(s)} D_{s,p} \times T_{s,p}) \quad (1)$$

where E = Total Energy Consumed

Z = Set of switches

P(s) = Set of Ports of switch s

F_s = Fixed Power Consumption of Switch s

T_s = Working Time of Switch s

$D_{s,p}$ = Dynamic Power Consumption of Port p of Switch s

$T_{s,p}$ = Working Time of Port p of Switch s

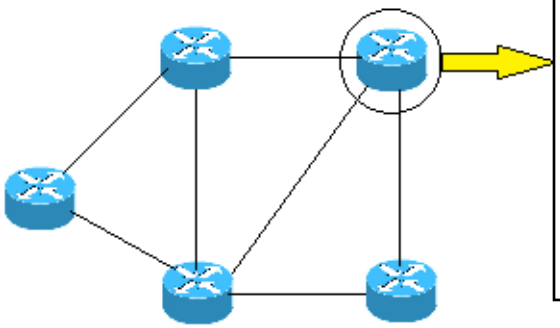
As we turn off more and more links i.e. ports on each side of a link, a larger amount of energy will be saved depending on for how long the links are being kept in sleep mode. Also, keeping in mind the connectivity constraints without incurring much loss and delays.

2.2. Software Defined Networking

Software Defined Networking (SDN) aims at providing flexibility and controlled or programmable application deployment in today's large-scale distributed computer networks. The basic motivation behind the development of SDN is to decouple the forwarding plane from the control plane as the network elements within traditional networks make both controlling as well as forwarding decisions in a distributed way. SDN architecture consists of three decoupled layers – management plane, control plane, and data plane. SDN control plane helps requests from the control applications to be translated to the data plane forwarding elements and provide control applications within management plane, an abstract network-wide view constituted by data plane elements.

Traditional network architectures are getting more and more ill-suited with the evolution of network technology and increasing link and port capacities, further making the management

of network processes more complex, not only at forwarding point of view but also at application and control level. Also, with the increasing cloud services and rising importance of Big Data, demand for bandwidth and other IT resources is increasing leading to a need for more intelligent and scalable network architecture with network-wide abstractions for better quality of services



Control Plane:

Populates Forwarding Tables and uses a large number of protocols like OSPF, IS-IS, BGP etc. to make forwarding decisions.

Data Plane:

Responsible for forwarding packets according to their local forwarding tables using a TCP/IP layered architecture.

Figure 2.1: Traditional Network

Network nodes in traditional networks, as shown in Figure 2.1 consist of both control plane as well as the data plane. Data plane is based on TCP/IP architecture which provides layered abstractions helping them to evolve rapidly, accommodating new technologies. In control plane, different control protocols are used to implement distributed algorithms populating the forwarding tables, based on non-local information received from Interior Gateway Protocol (IGP), making it very inflexible for rapid technology changes as compared to evolution in Data plane. As a result, the control protocols often draw up network topology, gather state information and perform node discovery to imply an intended logic on a particular topology. For example OSPF protocol collects information about network topology by exchanging messages with the OSPF running neighbors then applying the Dijkstra algorithm to compute shortest paths. So most of the time is spent to figure out the network topology and comply with the changes in the network topology.

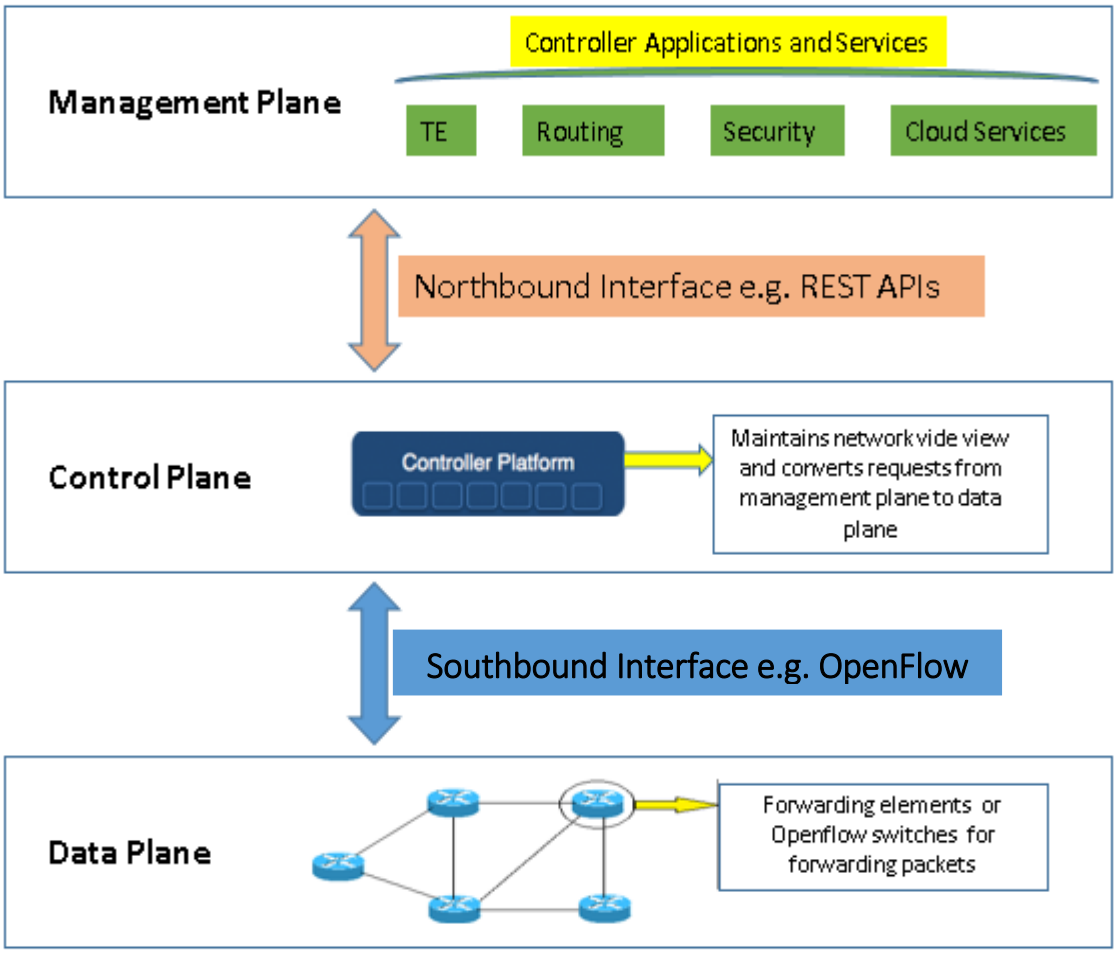


Figure 2.2: Software Defined Network Architecture

Software Defined Network architecture, as shown in Figure 2.2, concentrates on making the network more flexible and agile to support the advancing technology as well as storage infrastructure needs, using a centralized control over the network with a network-wide view. Making network controls programmable and providing an abstraction between forwarding network infrastructure and network applications. SDN approach centralizes network control by separating the control plane from the data plane and moving the control of the network to a centralized entity. Within, SDNs three decoupled layers i.e. the control plane, data plane, and the management plane. Management plane consists of Control applications, which interfaces with the control plane through Northbound Interface to observe and implement certain network behaviors using the abstract view of the network provided by the controller depending upon the user-defined programming, specified within the applications for internal decision-making purposes.

The control plane translate requests from management plane to the data plane elements and providing control applications in management plane with an abstract view of the network by interfacing with the data plane through Southbound interface. OpenFlow is the most commonly used open standard southbound interface to communicate between the control plane and the data plane. Control plane acts as an abstraction layer between the data plane and control applications maintaining a logically centralized intelligence of a network to create a consistent and updated view of the network. The Southbound interface provides communication between the data and control plane, updating control plane in case of event-based messages when there are changes within the network. Flow statistics sent by elements within data plane and packet-in messages sent to control plane in case the forwarding elements do not know what to do with the new incoming packets [2]. The Northbound interface allows management plane to leverage network services and capabilities without getting into details of the data plane infrastructure and managing traffic load as well as services within the network by directly programming the network elements, using a virtually centralized controller in the control plane.

2.2.1. SDN in Data Centers

Controller applications on top of SDN control plane, which can deploy and program custom network applications and services along with abstraction and logically central control of network has triggered the adaptation of SDN in data centers. Deployment of SDN within data centers can leverage large benefits by not only supporting scalability issues, due to increasing applications and services but also making them more efficient in the management point of view by cutting the Capital and Operational expenditures. There has already been a lot of research work done regarding deploying SDN architecture within data center networks to improve certain performance parameters or to support different applications and services within data center networks, which are complex to be efficiently deployed using traditional network architecture.

Within our research work, we have used SDN architecture to support the functioning of Segment Routing paradigm as it requires information regarding used bandwidth between the links and the global knowledge of the network traffic as perquisites to support segment routing

within the topology along with the network-wide view for helping routing decisions. In Section 3 and 4, we will be using two different approaches saving energy within the data center networks using the advantages of SDN architecture.

2.2.2. Energy Efficiency using SDN

SDN has been playing a big role in stabilizing growth of variety of networks, with quickly advancing technology having a big impact on improving the scalability and performance parameters. As we have discussed with the advent of technology, the data center networks, and service providers are struggling with the increasing energy spending and operational costs. Rodrigues et al. [24] propose a GreenSDN approach to emulate energy efficiency capabilities by creating an environment based upon Mininet and POX controller. As SDN can improve network operational parameters, with its centralized control and simple and flexible data plane, it is easy to deploy certain energy efficient traffic engineered routing plans using network-wide view along with the overall knowledge of different network parameters. Also, research works [25-27] used SDN architecture to support the implementation of SR by providing it information regarding bandwidth used between links and direct network-wide signaling to reroute traffic over backup or explicit paths.

In our research work, we have used SDN architecture to provide support for SR implementation along with per-packet based routing and Bandwidth Reservation, discussed in Chapter 3 and 4 respectively; with a motive to save energy and efficiently use available bandwidth.

2.3 Segment Routing - Its Need and Concept

There has been a lot of emphasis being put on developing extensions for routing protocols within a network to manage the traffic in an energy efficient manner. Classical IP routing techniques used for Energy reduction using machine learning techniques [28] provides a reduction in energy consumption in an intelligent manner by avoiding links to be chosen, which turned off before caused congestion. But it increases the IGP-TE flooding by 30%, which cannot be neglected due to the amount of delay produced as well as power spent on processing these floodings. Multi-Protocol Label Switching (MPLS) technology [29] can be

useful to cope with smart Traffic Engineering (TE) strategies, which handles the network resources in the most flexible way, and reacts dynamically to traffic changes. In the following Sub-Sections we have discussed the contribution of different technologies like MPLS and source routing, in the development of Segment Routing along with different energy efficiency concepts and features of SR.

2.3.1. Multi-Protocol Label Switching (MPLS)

In a network, there are certain set of Forwarding Equivalence Classes (FECs) deciding which packet belongs to a particular packet flow and what will be the path followed by it. In traditional IP forwarding, packet is re-examined at each hop and assigned to a new FEC, destined to a next hop until it reaches the destination. But in MPLS, the FEC assignment is done only once, at the ingress router. The FEC to which the packet is assigned, encodes as a short fixed length value known as a "label" [29], the header is not analyzed further, and the forwarding is done only by replacing the old label with a new label at each hop. Due to the concept of labels, Traffic engineering can be implemented very easily within an MPLS-enabled node. Each MPLS-TE enabled node supports both a routing protocol and a label distribution protocol. OSPF-TE [30] and ISIS-TE [31] are the possible routing protocols, which are the enhancement of traditional link-state based IGP i.e. OSPF and IS-IS respectively. The traditional IGP routing protocols have been enhanced with the ability to carry information regarding extended link attributes, providing a local constraint-based source routing. RSVP-TE [32] and CR-LDP [33] are the two commonly used TE capable label distribution protocols which are used to setup Label Switched Paths (LSPs), supporting both explicit route indication and reservation of resources during dynamic LSP setup. MPLS consists of three basic routing operations:

- Swap operation consists of swapping current label with a new label and forwarding the packet along the path associated with the new label.
- The push operation is used to push a new label on the top of existing label or pushing labels into an empty stack according to the path or services to be followed by a packet.
- The pop operation is used to pop out the active label to reveal inner label or exiting the MPLS tunnel in case of Penultimate Hop Popping (PHP).

2.3.2. Traffic Engineering in MPLS

The main objective of TE is to enhance the performance of an operational network at both the traffic as well as the resource level. The aspect of control within the traffic engineering can be proactive as well as reactive. Proactive TE can be the result of certain traffic requirements, generated on demand by the user or initiated due to change of resources available within the network. Reactive TE can be used to protect the network and maintain continuous traffic flow within a network, which may be triggered due to certain user-specified actions or redundancy requirements, in the case of failure. TE is one of the key aspects of MPLS, creating tunnels and providing constraint-based routing is one of the key features in MPLS-TE. Unidirectional tunnels are created on an ingress node, directed towards tail ends or egress nodes. MPLS-TE takes place in four steps [34]:

i. Link state protocols with extensions such as OSP-TE and ISIS-TE are used to carry the extended link attributes within their link-state advertisements (LSAs) or link-state packets (LSPs) throughout the network.
ii. A modified version of Shortest Path First (SPF) i.e. Constrained Shortest Path First (CSPF) is used to calculate the paths for particular flows according to the constraints defined by a user.
iii. If a path is available or meeting the constraints, Resource Reservation Protocol (RSVP) is used to signal the path and carry the label information.
iv. After signaling the path, the traffic is sent over the path depending upon the priority of different flows to be sent, and if data is to be sent in reverse direction, another tunnel has to be set up at the tail end as the tunnels are unidirectional.

2.3.3. Energy Efficiency using MPLS-TE

As we have discussed the advantages of MPLS and TE in improving the performance of core networks, there are significant number of research works focusing on saving energy consumption within core networks using MPLS-TE. Zhang et al. [35] use MPLS along with RSVP-TE, to manipulate the routing paths of a network so that the performance constraints, such as traffic demands can be satisfied using least number of routers. Energy can be saved by putting the idle links and routers to the sleep mode where the traffic demands are routed

through a set of pre-calculated shortest paths. Addis et al. [36] propose a model for minimizing the energy consumption within MPLS networks while handling the uncertain traffic demand changes. The results show that 50% of energy can be saved using MPLS routing protocol together with OSPF while handing traffic variations with very flexible TE approach.

Thus, MPLS with its Traffic Engineering property can be used to efficiently handle uncertain or on demand traffic changes within the network. As within our research work, we have used energy efficient routing algorithm causing a frequent traffic re-routing, as well as constraint-based traffic flowing through a certain set of links. MPLS technique forms a good base for us to dig deeper into our research work.

2.3.4. Segment Routing

Source routing is a process in which the path to be followed by a packet is already specified in its header. The intermediate network devices within the path will just forward the packet, according to the path decided by the source, assuming to have full knowledge of the network layout and best path-making decisions. Earlier, source routing was not used much due to certain limitations and security issues and network routing was more preferred, where intermediate network devices are supposed to have full network knowledge. But with the advent of technology and more and more emphasis being given to Software Defined Networking, implementation of source routing has become more feasible with fewer complications. With a network-wide view of the controller, which is one of the basic perquisites for Software Defined Networking, source routing can be easily used to reach certain networking and routing goals.

Segment routing uses the terminology of Source routing to forward the packets within the network, using the list of segments already specified within the header of a packet to route them on a pre-defined path. The SR header contains a pre-defined route encoded as a list of segments, a pointer to point to a particular segment of the list and the identification of the ingress and egress SR edge routers. The basic SR terminologies are given below, from which the whole SR process can be fathomed [37]:

- *Segment:* A segment identifies the type of instructions to be executed, either it can be the forwarding instruction to route a packet within a network or the service instruction, to provide specific services at a node.

- *SID:* Segment Identifier (SID) is used to identify a particular instruction or segment. It can be a Global SID, which is globally unique and is used to identify an instruction supported by all the SR-capable nodes within the domain or a Local SID, which is locally unique and is used to identify an instruction only supported by the node originating it.

- *Segment List:* Ordered list of segments specifying the source route of the packet along with service instructions to be executed at a particular node within the pre-defined path.

- *SRGB:* Segment Routing Global Block (SRGB) contains the set of globally unique SID's, from which we choose identifiers to represent particular instructions. Any segment outside the SRGB is of local significance.

- *IGP SID:* Segment identifier related to a piece of information, advertised by a link-state IGP e.g. IGP-Prefix, IGP-Adjacency, and IGP-Node SID.

- *Prefix-SID:* It is a globally unique SID within SR domain, used to identify the ECMP-aware shortest path, computed by IGP to the related prefix.

- *Node-SID*: It is a globally unique IGP Prefix-SID used to identify a specific router.

- *Adjacency-SID:* It is used to identify a unidirectional adjacency or a set of unidirectional adjacencies, and it is local to the node advertising it.

- *Anycast-SID:* It is an IGP-Prefix-SID, used to identify a specific group of routers and is globally unique.

- *SRDB*: Segment Routing Database contains a set of entries identified by a segment value. The instruction associated with each entry at least defines the identity of the next-hop, to which the packet should be forwarded and the operation which should be performed on the SR header.

- SR-header operations: There are three SR-header operations [37].
  i) PUSH: An SR header is pushed on an IP packet, or additional segments are added at the head of SR list, and the pointer is moved to the first entry of segment list.
  ii) NEXT: The current segment instruction is completed, and the pointer is moved to the next segment within the list.
  iii) CONTINUE: The current segment instruction is not completed, and the pointer is left unchanged.

As above we have discussed the basic terminologies regarding Segment Routing, basic SR routing operations are discussed below:

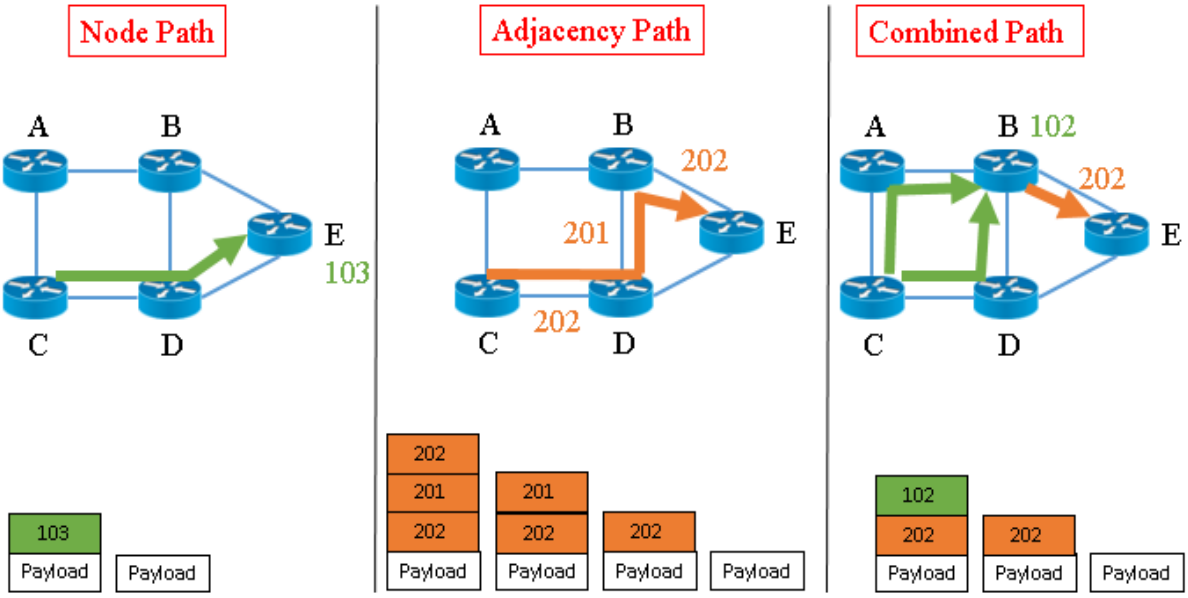


Figure 2.3: SR Routing Operation [38]

Figure 2.3, shows different paths being used with the help of different SIDs, to send packet from Ingress node i.e. node C to Egress node i.e. node E.

- *Node Path:* In node path, only a single globally unique Node-SID is used i.e. 103, which is associated with Node E i.e. destination node. At node C, PUSH operation is

used to push the segment list into the header and will CONTINUE to node D, which is the shortest path to reach Node E, learned through IGP. Node D will check its SRDB, corresponding to the segment 103 and then CONTINUE to node E. As node E is destination node, so penultimate hop popping will take place at node D.

- *Adjacency Path:* In adjacency path, locally significant Adjacency-SID's associated with each Adjacency are used, hop by hop, in an ordered manner. At node C, PUSH operation takes place pushing segment list into SR header of the packet and packet is forwarded to node D according to the routing instruction associated with SID 202. At node D, the pointer moves to SID 201 and node check its SRDB and forward the packet to node B, moving the pointer to SID 202. As adjacency-SID are locally significant to the node, so node C and D can have same adjacency-SID i.e. 202 associated with it. Node B further pops out the segment and sends it to the destination node according to the instruction associated with SID 202.
- *Combined Path:* Combination of different SID types can be used to reach our destination, e.g. in above fig., node-SID 102 is used to reach node B. Assuming there are two equal cost paths available to node B, it uses ECMP to load balance and reach at Node B, as discussed in the Node Path section. At node B, adjacency-SID 202 are used to reach our destination.

2.3.5. Traffic Engineering using Segment Routing

Traffic Engineering is one of the most discussed use cases in the application of Segment Routing. This source-based routing model allows traffic engineering to be implemented without the need of a signaling protocols. IGP protocol extensions i.e. OSPF-TE and ISIS-TE can be used for the signaling of the explicit paths on the base of different requirements and modes. RSVP-TE is one of the major components in the SR-TE architecture, which is used to signal and establish explicit paths. Traffic engineering within the SR can be classified into two use-cases [39]:

i. Traffic engineering without bandwidth admission control:

- *Dual plane networks:* SR traffic engineering can be used to provide Disjointness between planes within a network, based on a dual plane design. Traffic can be load

balanced using ECMP within the whole network or a specific plane of the network depending upon the constraints provided by the user.

- *Egress Peering Traffic Engineering:* SPRING based traffic engineering should allow any ingress node to select an egress node or the exit point of the packet, using various explicit paths or specific nodes providing a particular service depending upon the constraints provided by the user, at the ingress.
- *Load balancing among non-parallel links:* SR based traffic engineering architecture should allow the traffic to be load balanced between nonparallel links, leading to different neighbors.

ii. Traffic engineering with bandwidth admission control:

- *Capacity Planning process:* It consists of simulating the placement of the traffic, along ECMP-aware shortest-paths and accounting for the resulting bandwidth usage, where bandwidth is calculated by any planning device or PCE server. If congestion is produced due to increase in the bandwidth demands, a capacity increase is triggered, failure of which can lead to a traffic engineering policy to be initiated, which will route the specific amount of minimum traffic on explicit paths to avoid congestion.

2.3.6. Path Protection using Segment Routing

A network supporting segment routing should deploy different Fast Reroute (FRR) techniques to cope with the node or link failures using pre-computed backup paths. Path protection in SR can be classified into two cases [40]:

i. Management free Local protection:

- *Management free bypass protection:* Providing local protection by enforcing the traffic along the shortest path from the failed node or link, reconverging at the protected next hop in the case of link failure and next-next hop in the case of node failure.
- *Management free shortest path protection:* Providing local protection by pushing the traffic on the shortest path towards destination from the point of local repair (PLR), instead of merging back on the previous path.

ii. Managed Local protection:

- In managed protection, backup paths can be provided both by Bypass or shortest path protection ways .The only difference from management free protection is that here, the user can specify high-level constraints to follow certain explicit configurations like minimizing the metric cost, end at next hop or next-next hop, etc. for a backup path.

2.3.7. Segment Routing using MPLS Data Plane

The most attractive advantage of Segment routing for using it within existing networks is that it can be leveraged over existing MPLS data plane. Although SR does not rely on Label Distribution Protocols (LDP), if present, they can coexist with them. In the MPLS instantiation of SR, the following applies [41]:

- The list of segments is represented as a stack of labels.
- The CONTINUE operation of SR is implemented as an MPLS swap operation.
- The NEXT operation is implemented as MPLS pop operation.
- The PUSH operation is implemented as MPLS push operation for pushing a stack of labels.

In a nutshell, SR presents an improved version of MPLS, as it can totally coexist with different MPLS protocols within MPLS data plane with advantages and applications far better and scalable within today's complex networks, co-operating with the increasing demand of core and service provider networks.

2.3.8. Segment Routing in Data Centers

Data Center networks consist of highly symmetric topologies with multiple parallel paths between two server points or nodes. The Clos topology is the most used data center network topology, among the operators. Traffic can be load shared between multiple paths using ECMP mechanism, but there are certain drawbacks within the data centers, which have become the points of discussion and improvement in latest research works [42]:

- ECMP within the data center is per-flow based, making it difficult to perform load sharing for elephant flows, reducing the performance of mice flows completion and leading to frequent congestion on a particular path or set of links.

- Due to the lack of determinism, if a path failure occurs, a new connection with a different path may get selected between two hosts. Leading to a complex rerouting of a failure which increases linearly with the number of parallel paths within the network.
- Remembering, the bad paths within the network based on previous collisions and congestion makes it easy for transport protocols to take more efficient path making decisions. But Transport protocols like TCP are not able to remember bad paths. Thus, every new connection is oblivious to paths chosen before or by other nodes.

Although, if we are using source routing based scheme, like segment routing on a data center network with MPLS data plane, ingress nodes or hosts will specify a pre-computed path on the packet header pre-determining its end-to-end path and removing the obliviousness of the path to be followed. Also, using a central controller or a PCE server with a network-wide view, we can make routing decisions based on the history of congestion and collisions within the network, ultimately improving the performance of the data center. Also, with the ability to choose a path at the ingress, we can select different segment routing instructions based on not only per-flow but also on per-packet or flowlet level, making it easy to load balance the elephant flows along different parallel paths, keeping in view the flow completion time of short flows.

The large-scale data centers use BGP as a common routing protocol, to provide end to end connectivity along with ECMP mechanism for load balancing. While applying Segment Routing in Data center with MPLS, data plane uses BGP Prefix segment, which is network wide instruction to forward the packet to the prefix along a ECMP aware best path without any need of label distribution protocols.

2.3.9. Energy Efficiency using Segment Routing

In the above section, as we have discussed the role and advantages of segment routing in the data center networks, even applying it to MPLS data plane leverages huge benefits. But not much research work has been done to fully explore the advantages of deploying this source routing based scheme. Energy efficiency is one possible area where segment routing can have a significant impact as there is some research work already been done using MPLS-TE, as discussed in Section 2.3.3. Although with the prospect of vast possibilities in saving energy,

in our best knowledge very few research initiatives [25-27] are related to energy efficiency using Segment Routing.

Carpa et al. [25], use a STREETE framework to save energy within the backbone networks by dynamically adapting some turned off links to the traffic load. For performing efficient rerouting and declaring explicit paths, segment routing is used within an SDN framework. The results were evaluated by using real networks with real traffic matrices using OMNET++ simulator, resulting in 44% of links to be turned off while maintaining the quality of service. Although the results were significant, criteria for selecting a link to be turned off did not consider any bandwidth constraints. Also, not much focus was given on keeping the traffic within the available links, as all links were supposed to be turned on if any single link gets congested. The reason being, turning on links one by one is far more complex than turning them off. The authors tried improving the algorithm [26] for managing load surges within a network, avoiding turning on unneeded links. Lee et al. [27], proposed an routing algorithm, using SR on MPLS data plane, specifically, oriented towards improving average network throughput and average rejection rate of routing requests by balancing traffic load and cutting the extra cost of packet header size. But not much emphasis was given on managing the traffic load surges or criteria for turning on the links, in the case of congestion.

2.3.10. Segment Routing using SDN:

SDN architecture can be applied to different frameworks like service provider networks, enterprise data centers or large carrier networks. Segment Routing is related to a centralized control plane implementation, where a dedicated controller keeps a network-wide view and performs path computations and encoding paths using a sequence of SIDs. For the implementation of SR, we need certain perquisites about knowledge of different network parameters like link capacities, congestion or bad path logs and network-wide view. SDN using three decoupled layers i.e. control plane, data plane, and management plane, can provide all these services and environment for an actual realization of source routing based segment routing paradigm on a network. With the help of SDN controller, SR can provide help in building on-demand or on per-flow based, traffic engineered tunnels using information and programmability provided by a centralized controller. Also, segment routing simplifies the

implementation of control plane and reduces the requirement to maintain intermediate node state, which further facilitates the SDN implementation.

Sgambelluri et al. [43], used a specifically enhanced version of the OpenFlow SDN controller along with OpenFlow switches, to support the SR implementation within the network to successfully demonstrate the SR based source routing and dynamic traffic rerouting, without any use of signaling protocols or any packet loss. Ethernet technology with its increasing importance in WAN along with its compatibility with SDN implementation, especially within service provider networks has led to many research initiatives [44, 45]. Bidkar et al. [44] propose and simulate a field trail for an SDN implementation using Carrier Ethernet and SR technology, where Carrier Ethernet provides a programmable control plane along with the source routing capabilities provided by SR, resulting in higher bandwidth efficiency and improved availability of the network. Cai et al. [45], propose a new carrier Ethernet architecture, using the combination of fully distributed unified MPLS network architecture and fully centralized OpenFlow-based SDN architecture, by combining their individual advantages to simplify and increase the performance of a network. Davoli et al. [46], develop an SDN-based, OSHI architecture to provide traffic engineering with segment routing, using hybrid IP/SDN nodes. The SDN controller along with SR daemon present within each hybrid node is used to provide an edge to routing services with better performance and shorter SR path lengths.

2.4. Per-packet Load Balancing in Data Centers:

In data centers, the communication takes place as flows, where flow is defined as a sequence of packets being transferred between a particular source and destination. Short flows are few kilobytes in size and are related to user's tasks, which generate requests and responses between servers. Long flows have a size of several megabytes, and they carry the data for the operation and maintenance of the data center. Small Flow Completion Time (FCT) for short flows and high throughput for long flows simultaneously achieved is the key to High-performance data centers [47]. Congestion within the network can impair the simultaneous achievement of short FCT and high throughput, but with the availability of multiple paths within data center networks, link or path congestion can be bypassed. There are different multipath routing

protocols already in use within data centers like ECMP [48] being most commonly used. Also, Flow-based load balancing mechanism used to load balance traffic load among equal cost multiple paths and avoid congestion. Flow based load balancing mechanism does not differentiate between long flows and short flows, as both types of flows can be routed on the same path, leading to large queueing delays or congestion within delay sensitive short flows. One of the solutions to solve the problems of flow based load balancing, as discussed in IETFs internet draft of SR [42], is per-packet load balancing. Per-packet load balancing independently forward every incoming packet on a randomly selected output port, considering in advance knowledge of multiple equal cost paths between source and destination. With the ability to choose paths on the host, one may go from per-flow load sharing in the network to per-packet or per-flowlet based load balancing [42].

Some per-packet load balancing approaches are already proposed, like DeTail [49], Random Packet Spraying (RPS) [50], and Multipath TCP (MPTCP) [51]. DeTail propose to use per-packet adaptive load balancing approach, along with a reorder-resistant TCP to prevent it from reacting out-of-order packets, as an indication of congestion aiming at reducing FCT of short flows. MPTCP split a TCP flow into multiple sub-flows and forwards them to a destination on multiple paths using ECMP, avoiding the complexity of transmitting a long flow through a single path along with multiple short flows. Whereas, RPS use packets of the flow and forwards them over equal cost multiple paths, where each path for a packet is randomly selected with uniform probability, without any modification at switches or servers. With the introduction of SDN paradigm within data center networks, per-packet load balancing can be easily realized using the centralized network control and easy programming of network devices provided by SDN controller, ultimately increasing the performance of data center networks.

Chapter 3: Per-packet Based Energy Aware Segment Routing Model for Data Center Networks

In this chapter, we have proposed and simulated a mechanism to reduce energy consumption within data center networks using segment routing (SR) supported by SDN architecture and per-packet load balancing. SDN controller is able to provide prerequisite which are required for the implementation of segment routing paradigm within a network along with SR supporting nodes. The complexity of handling every flow at a packet level to support per-packet based load balancing is made easy using network-wide view and knowledge of equal cost paths by SDN controller, along with the help of source routing based SR. Per-packet load balancing is basically used to avoid congestion between long and short flows, which the flow based load balancing techniques like ECMP are not able to tackle efficiently.

The basic aim is to keep the traffic within a particular set of available links as long as traffic load is below a particular threshold, which if surpassed could lead to congestion. Then, putting the unused links into sleep state so that we can reduce the energy consumed by the data center networks.

The remainder of the chapter is organized as follows: Section 3.1 discusses the design principles of our approach. Section 3.2 describes the architecture of our energy efficient approach. Section 3.3 lists the details of our approach's operation. Section 3.4 covers the implementation details of our approach. Section 3.5 lists the conditions and constraints for implementing this approach. Section 3.6 covers the experimental setup and the evaluation results. Section 3.7 concludes the chapter.

3.1. Design Principles

As discussed earlier about some already proposed energy efficiency research works [33, 35] using segment routing which particularly focuses on turning off as many links as possible. Turning off the links and rerouting the whole traffic on available links is one thing and keeping the traffic on available links, for as long as possible without congestion is another, as these both are dependent upon each other. So we have focussed on both the aspects and introduced a per-packet based energy efficient segment routing approach which leverages the advantages

of SDN mechanism along with the simplicity of source routing based SR to save energy within data center networks. Our goal is achieved by turning off unused links and keeping the traffic load on a particular set of nodes as long as a particular threshold is not reached. The traffic is load balanced between multiples paths using per-packet load balancing to well manage the traffic between active links and to avoid frequent congestions. Threshold, when surpassed on a single active link, will lead to turning on all the inactive links and again the process of turning off particular links is started by selecting links to be turned off, using our proposed algorithm.

Deployment friendliness: While introducing a new approach, one of the main concerns is how easily it can be deployed within actual network environment. As we have discussed in Section 2.3.7, Segment routing can be easily leveraged on MPLS dataplane, which is commonly used in today's data center networks. The per-packet load balancing approach can also be easily deployed by making programming changes within the SDN controller and SR supporting nodes where every node randomly selects one of equal cost multiple paths to forward a packet. Although an explicit path can also be specified for packet forwarding by making changes into pre-determined source routed path to select a particular link out of multiple equal cost paths, using Adjacency-SID's. The deployment of our approach does not require any hardware or software changes to MPLS dataplane of an SDN based data center network.

Also, data center networks provide a good environment for the deployment of our energy efficiency approach as SDN architecture based data center networks are easy to manage and scale, using a single administrative control. Using central controller it becomes easy to make decisions regarding which nodes are to be turned off and how the traffic is to be routed over active links using local rerouting or explicit paths. Per-packet load balancing also leverages the availability of multiple paths between two particular nodes in a typical data center network like Fat-tree or Clos network topology making it possible to efficiently load balance the traffic from one point to another.

3.2. Architecture

Our approach is based upon using an SDN controller to program the nodes in MPLS dataplane to provide segment routing as well as per-packet load balancing between links. The dataplane

within our architecture can use IGP i.e. OSPF-TE or ISIS-TE to distribute topology information within a traffic engineered topology keeping in regard bandwidth and other administrative constraints. The SR configured MPLS dataplane helps to support source routing and forwarding packets based on labels. The SDN controller provides a network wide view gathering the information regarding certain traffic attributes and parameters to support segment routing and per-packet load balancing within SR supporting nodes in the dataplane. Certain Application Program Interfaces (API's) at the top of the SDN controller are used to program the dataplane and controller as shown in Figure 3.1, to efficiently turn off as many links as possible. The details regarding each layer of architecture is given below:

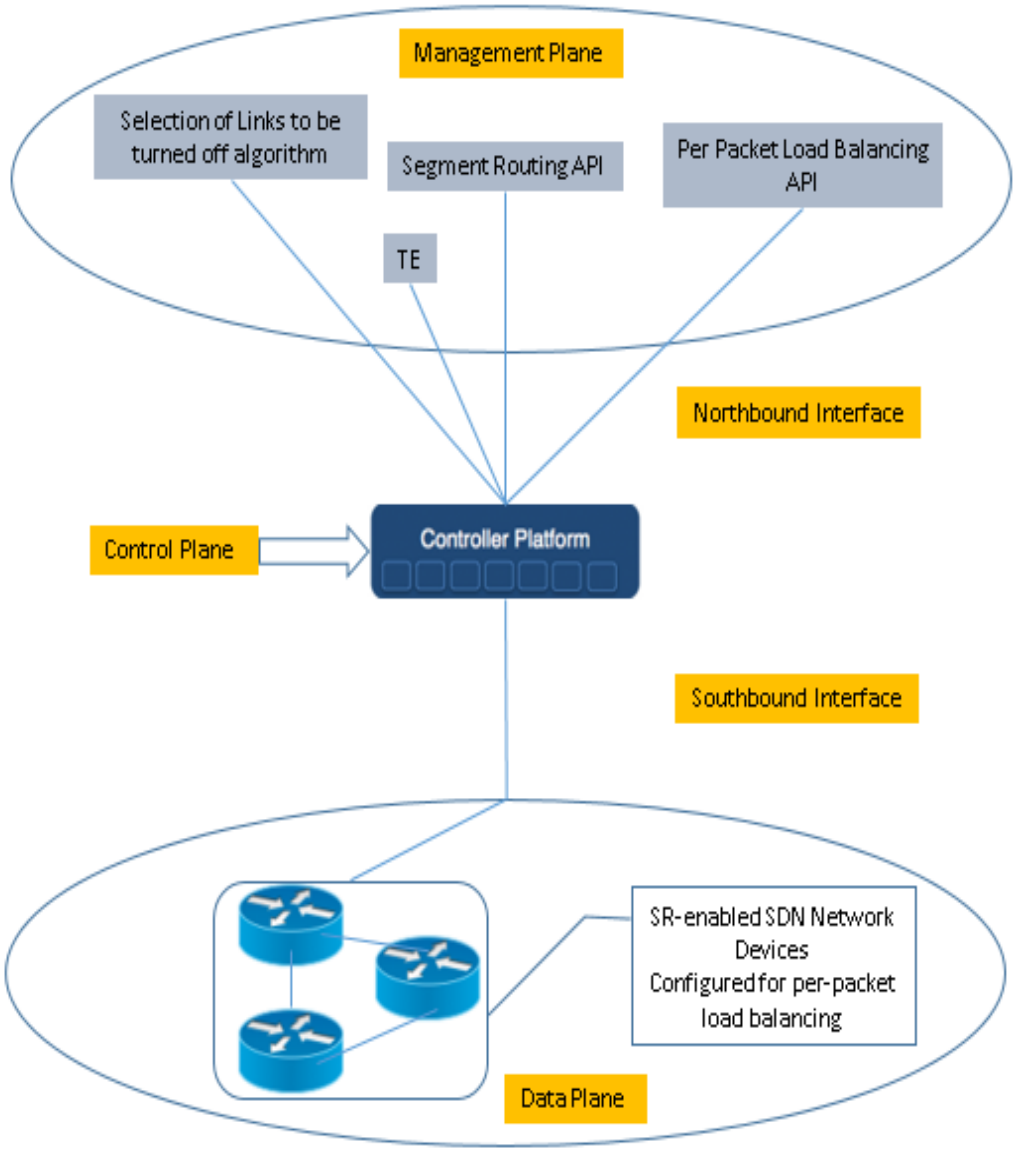


Figure 3.1: Architecture

a) **Management Plane:** It consists of different protocols and API's which are used to directly program the SDN controller as well as the SDN network devices according to the functionality required. In our architecture, we have programmed controller and nodes according to the algorithm for selecting the links to be turned off along with segment routing and per-packet load balancing API's to provide certain protocols and program the network elements, for the implementation of our energy efficient approach.

b) **Control Plane:** It is used to keep a network wide view of the network along with Forward Information Base (FIB) table to help network elements with forwarding of packets they are not able to handle and Forward Equivalency Class (FEC) table to decide what actions are to be done on the classified packets. It also helps control applications to directly program the network elements according to the user requirements.

c) **Data Plane:** It consists of data center network topology, which further consists of SR enabled SDN network elements to perform SR based source routing using MPLS dataplane. Also, providing the functionality of per-packet load balancing according to the network information provided by the SDN controller. A Node Controller is attached to each SR-enabled Label Switched Router (LSR) which populates the local forwarding tables, waits for commands from SDN controller and keeps a global view of the network. Also, an IGP protocol is used to discover neighbours.

3.3. Operation

The operation of our energy efficiency approach can be divided into three phases which include the decision of selecting the links to be turned off using the proposed algorithm. Deciding the rerouting paths for the traffic to be forwarded on the path that are to be turned off and actually turning off the links using the signalling messages between the nodes and the controller. The whole operation takes place within the SDN architecture where SDN controller gathers all the information regarding the network topology and convert requests from management plane applications into data plane configurations. Role and operation of each of the phase is discussed below:

1. **Selection of Links to be turned off:** A typical data center network consists of multiple paths between an Ingress-Egress Pair (IEP) making it possible to efficiently load balance the packet between available multiple paths. Although, in normal scenarios, not all the links are used leading to the wastage of energy. In this phase, we will use our proposed algorithm to select a particular set of links to be turned off, first between multiple links within all IEP pairs in the network and then selecting the common links which can be turned off. Along with, keeping in mind the connectivity constraint and leveraging the information about network metrics and current as well as past traffic pattern of the network links from an SDN controller. We will further discuss the details of Link Selection algorithm in Section 3.4.
2. **Deciding rerouting paths:** The links to be turned off are already computed by the SDN controller using the link selection algorithm. Now before we actually turn off the links, we need to decide and reroute the traffic scheduled to be forwarded on turned off links towards the set of available links. Congestion can come into picture, if we let each node to recalculate new shortest paths toward other nodes. Thus, SDN controller is used to compute new routes by exploiting its global knowledge about the network traffic. OpenFlow SDN specifications already consider the availability of knowledge about bandwidth usage between any head end and tail end, which is the basic prerequisite to reroute the traffic. The source routing based segment routing helps in making the rerouting process simple by replacing the current SIDs with node-SIDs associated with available links. An explicit path can also be selected while rerouting the traffic over a particular link using Adjacency-SIDs. SDN controller monitors the bandwidth utilization within all the available links and if any of the links crosses a pre-defined threshold limit of traffic load, all the links are turned on again and the traffic is routed over to their original shortest path routes.
3. **Actual Rerouting and turning links on or off:** Using the list of links to be turned off calculated using phase 1 and rerouting paths computed in phase 2, we can start turning off links using the SDN controller which further uses signalling messages and certain protocols to turn off the links within the network. The flowchart for the whole rerouting and turning off links is described in Figure 3.2. Initially, SDN Controller informs all the routers within the data center network about the pre-computed reroute paths for

each link to be turned off. As the controller is having the list of links to be turned off, so it informs nodes adjacent to those links to turn them off and wait for an acknowledgment from the nodes. As soon as the nodes adjacent to links to be turned off acknowledge, the controller sends an IGP flooding that the links to be put into sleeping mode are down. Every router make changes to their FIB according to the pre-computed reroute paths provided by SDN controller, to reroute the traffic onto available links. After the traffic is rerouted, nodes adjacent to links to be turned off suspend HELLO messages to completely put links into sleep mode.

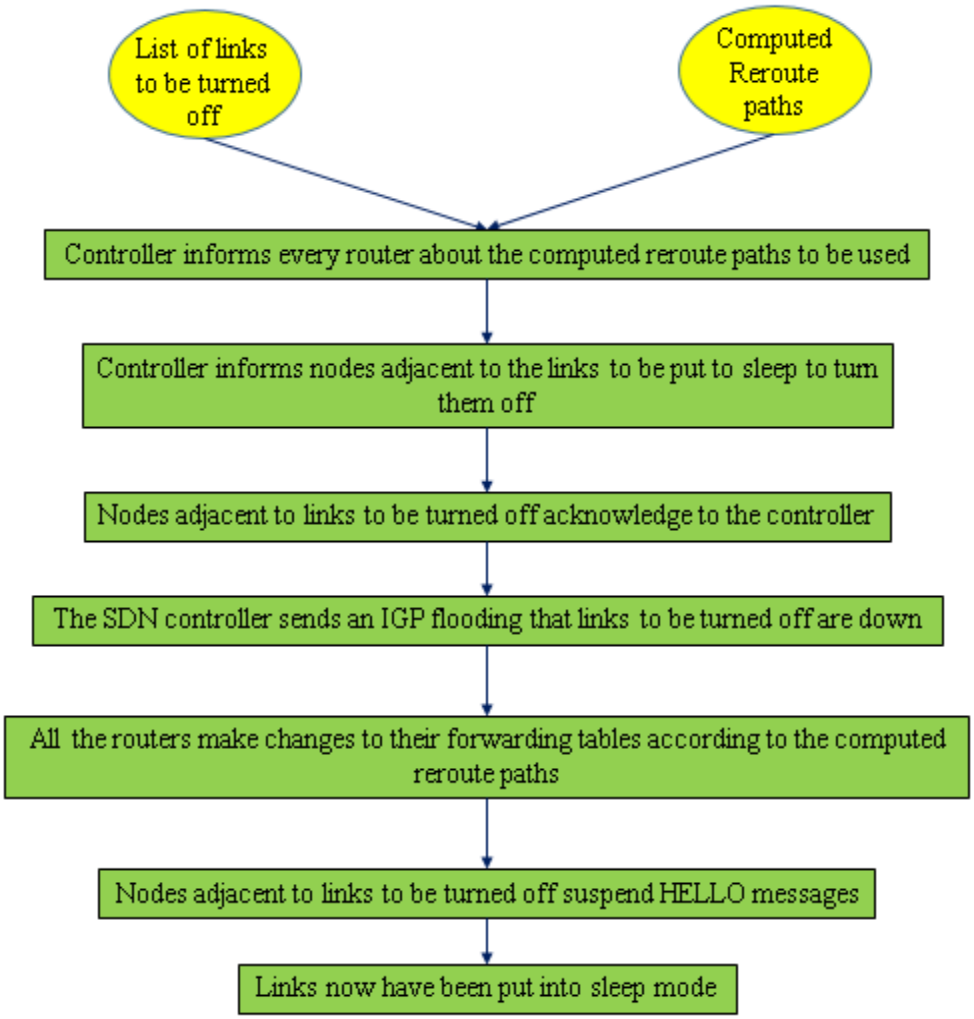


Figure 3.2: Actual Rerouting and Turning Off Links

SDN controller is used to continuously monitor the bandwidth utilization over all the active links and if the bandwidth utilization crosses a predefined threshold value, all the

inactive links are turned on again and traffic is routed over their original shortest path routes The reason for immediately turning on all the links instead of turning them on one by one to avoid congestion, is discussed in detail in Section 3.6. RPS [45] is used to randomly forward packets within available equal cost multi paths instead of per-flow based ECMP which is not able to differentiate between long and short flows, so that congestion can be avoided as long as possible within available links.

3.4. Link Selection Algorithm

As discussed in the previous section, the SDN controller is used to make decision of selecting a set of links to be turned off within a data center network using protocols provided by our proposed algorithm, which directly configures the SDN controller to gather certain information from data center network. The prerequisites for implementing our proposed algorithm includes knowledge of all the links within the network along with the list of multiple paths between each set of IEP i.e. L(n), which is the responsibility of SDN controller to gather. SDN controller also gathers information regarding bandwidth utilization and capacity available between each link and number of hops within each path, required for the implementation of our proposed algorithm. Using the information gathered from the network, SDN controller will calculate Cumulative capacity of each path (C) and Cumulative energy of each path (E) within an IEP.

Now, within an IEP the path with the highest number of hop count is selected because initially traffic will be routed on the paths based upon shortest path algorithm and paths with higher hop count or longer paths will have lower traffic load. Also, we are targeting on turning off underutilized links of a particular IEP. If the traffic load (L) within the IEP is smaller than $(C\text{-}c_n)$, where c_n is the capacity of path n, the selected path is not added to the list of paths required between an IEP i.e. L (M, c) and it remains active. When path stays active, there becomes no change either in capacity or in energy consumption for that particular IEP. Otherwise, if the cumulative capacity of an IEP after removing a particular path capacity is less than L, the link is added to L (M, c) and it is supposed to be turned off. Also, if a selected path is closed, both capacity and energy consumption decrease by $(C\text{-}c_n)$ and $(E\text{-}e_k)$,

respectively. After following this logic loop on all the paths within the IEP, we will receive the L (M, c) for a particular IEP.

Algorithm 3.1: Calculating list of Paths required

INPUT: **<** L (n), C, E, c_n **>**

OUTPUT: **<** L (M, c) **>**

for L (n) while L (n) ≠ Empty

k= path with maximum number of hops in L (n)

// if traffic load is greater than or equal to the capacity of IEP (removing path k capacity)

if (C-c_k ≤ L)

L (n, C) = L (n-1, C)

Remove path k from list L (n)

// if traffic load is smaller than the capacity of IEP (after removing path k capacity)

else

L (n, C) = L (n-1, C-c_k) - e_k

Remove path k from list L (n)

// making decision about particular path k

end if

// making decision about all the paths between an IEP

end for

// return the list of paths required between an IEP

return L (M, c)

Table 2: List of Notations

NOTATION	EXPLANATION
L(n)	List of Multiple Paths between an IEP, where n= (0, 1, 2…... N)
C	Cumulative capacity of each path between IEP
E	Cumulative energy of each path between IEP
$\mathbf{c_n}$	Capacity of path n, where n= (0, 1, 2…... N)
L (M, c)	Return the list of paths required between an IEP
L	Total load between an IEP
$\mathbf{e_k}$	Energy consumed by link k
L (R,$\mathbf{c_R}$)	List of paths which are not required between an IEP
$\mathbf{l_{off}}$	Links to be turned off in that particular IEP

Using L (M, c) and List of multiple paths between an IEP, we can calculate the list of links to be turned off between that particular IEP using the algorithm below:

Algorithm 3.2: Calculating links to Switch Off

INPUT: **<** L (M, c), L (n) **>**

OUTPUT: **<** l_{off} **>**

for L (M,c) ≠ Empty

// calculate the paths not required between an IEP

L (R,c_R) = L (n) - L (M, c)

//calculate the links to be turned off in an IEP

l_{off}**= links{** ***L (R,*** c_R***)} – [links(*** ***L (R,*** c_R***)}*** ∩ ***links{ L (M, c)}]***

end for

SDN controller will calculate the list of paths which are not required i.e. L (R,c_R) by subtracting L (M, c) from L (n). There may be a case if a particular link of a path in L (M, c)

is present in path of list L (R,c_R). An intersection between links of both the lists, links{L (R,c_R)} ∩ links{L (M, c} is used to point out common links and are subtracted from the list of links that are not required. SDN controller will follow the whole process above for multi-paths of all the possible IEPs within the network. After calculating the list of paths that are not required for all IEPs, if any of the links is used by some other path that is required, the link is not added to l_{off}. Otherwise, links within all the unrequired paths that are not used by any required path are used to populate l_{off}, keeping in concern the connectivity constraint so that connection between an IEP is not lost. Further, this list along with the pre-computed rerouting paths is used to actually put the links into the sleep mode and route them on to available links.

3.5. Implementation Details

We have implemented our approach by using OMNET++ 5.0 INET framework, where SR module is implemented over top of INET's MPLS module by making changes to support source routing forwarding using segment identifiers. The implementation corresponds to the theoretical high-level definition of segment routing/SPRING, where the packets are not using the MPLS and IPv6 format but a list of segment identifiers (labels). Also, we have only implemented adjacency-SID and node-SID from segment routing RFC's, leaving anycast-SID or any other SIDs. A simplified SDN control plane is used instead of any standardized OpenFlow SDN approach which is actually used to reroute the traffic using tables or explicitly monitor the traffic and bandwidth utilization within links. Also, a *node controller* is used along with each LSR switch, to make the SDN switches within network more powerful. Node controllers are used to discover neighbours, populate forwarding tables and wait for commands from the controller. Each of the nodes within the network is configured to randomly select an output port within available multiple equal cost paths to provide per-packet based routing depending upon bandwidth utilization of each node.

3.6. Conditions and Constrains

SDN controller exploits global knowledge of the network traffic to help the implementation of our algorithm, where initially it is assumed that the SDN controller should be aware of the traffic matrix between any IEP for deciding links to be turned off and further deciding

rerouting paths. Fortunately, this kind of availability is already included in the OpenFlow SDN specifications. The buffer size for each of the switch port is assumed to be same, so as to buffer the packets for per-packet load balancing. But, the introduction of per-packet load balancing approach leads to out of order packets constraint. Out of packets are not a problem in per-flow based routing but in per-packet based approach due to the delay over each of the link, which varies depending upon the traffic load over that specific link, leads to packets getting out of order. Our approach can have a significant adverse effect over the Voice over IP (VoIP) messages but despite this it is advantageous over improving Flow Completion Time (FCT) and throughput of flows within a data center network, ultimately increasing the sleeping time of links. Thus, per-packet approach is advantageous in certain applications where Priority of getting higher throughput and lower FCT is more as compared to in-order packets.

Also, we have considered turning on all the links if traffic load on any single available link crosses a pre-defined threshold. There may be a possibility that come into picture, that why not we turn on links one by one to avoid the congestion as compared to turning on all the links at once. But the decision regarding turning on links is very difficult to make as compared to turning them off. A particular link that we may decide to turn on, it may be the shortest path for some other links. Further, leading to more traffic being routed on that path, eventually leading to more severe congestion. Although there has been work done focusing on carefully turning on links one by one [34], but it is out of our research scope due to complexity issues.

3.7. Evaluation and Analysis

This section includes the details regarding emulation based experimental tests and the criteria used to evaluate our algorithm and approach. The section also covers the results of the evaluation tests and the discussion about the same.

3.7.1. Experimental Setup

Details about the OMNET++ based experimental setup and the INET framework traffic generator, used for evaluation of our approach are incorporated here along with the test topology, different test parameters and workloads used in our implementation.

3.7.1.1. OMNET++ Based Experimental Network

OMNET++ is widely being used as a network simulation platform which uses extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators [47]. OMNET++ also includes Eclipse based-IDE which is used to create and configure models and build up the simulations using the inbuilt or self-defined libraries. NED editor in OMNET++ is used to create and program compound modules, channels and other component types, creating a virtual topology with different components. Also, ini File editor is used to configure simulation models for execution to support different module configurations. The simulations can be launched directly from the IDE as a normal C/C++ application and perform C++ source-level debugging on it. Though we can also run it as a standalone application (under Tkenv or Cmdenv) or run batches of simulations where the runs differ in module parameter settings.

In our simulations, we have used OMNET++ 5.0 Eclipse IDE to configure our SDN controller as well as Node controller modules, using C++ coding at different network components within NED editor and ini File editor. MPLS module of INET frame is used to create a segment routing module, which closely follows the libraries and code of already provided MPLS module to support segment identifiers based source routing. Further, the SPRING/SR based network includes Node controller and Network controller modules along with nodes configured for per-packet based Random Packet Spraying within equal cost multiple paths to provide our required simulation scenario. OMNET++ is well suited for performing data center network simulations and experiment on networks that are constrained by network properties such as bandwidth, latency, and queueing. Tests are performed using OMNET++ running on Windows desktop with the following configuration - Intel Core i5 with two 2.2 GHz cores and 12 GB of RAM.

3.7.1.2. INET Traffic Generator

In our simulations the traffic generation is performed by INET *UDP BasicApp* traffic generation module which is already available in the OMNET++ INET framework. We can generate constant bit rate or variable bit rate traffic depending upon the settings of the *sendInterval* or equivalent parameters. In our simulation we have used the UDP videostream

client server application to generate traffic and further its parameters can be changed by using different parameters within its ini file. Below are the basic example commands in the .ini file of our topology:

**.udpApp [0].typename = "UDPVideoStreamSvr"*
**.udpApp [0].localPort = 1234*
**.udpApp [0].sendInterval = 1s*
**.udpApp [0].packetLen = 1000B*
**.udpApp [0].videoSize = 10MB*

The commands will generate the traffic within the data center network, where ini File commands can be further used to change the traffic parameters according to the requirements. The above commands specify the type of traffic being used, the local port from where the traffic will be sent, the time interval between the packets, the packet size and the size of the video being sent respectively.

3.7.1.3. Test Topology and Parameter Settings

As our aim is to reduce the energy consumption within data center network, so we have used a 4-ary Fat tree topology for our simulations.

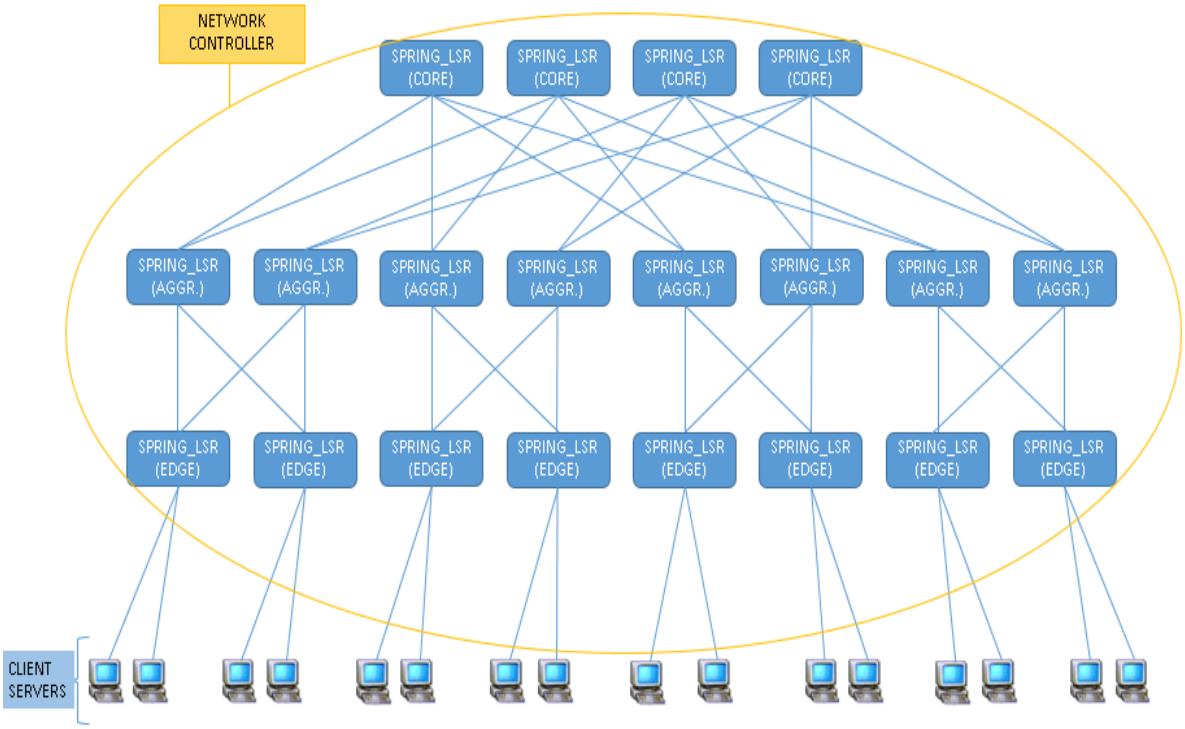


Figure 3.3: 4-ary Fat Tree Topology for Experimental Evaluations

It is basically a three layer topology which includes eight edge, eight aggregation and four core switches along with 16 client servers. Fat tree topology also supports our requirement of multiple paths between any two switch pair or IEP, as each core switch connects to four aggregation switches, each aggregation switch is connected to two edge switches and two core switches. Also, each edge switch further connects to two servers providing multiple paths. Each of the SPRING_LSR consists of a node controller as discussed in Section 3.5 which is connected to the network controller and makes the LSR's within the network more powerful to support certain routing decisions. The node controller is not shown in topology because it is a module which is internal to all the nodes and basically provides a two-way connection to the controller.

Table 3: Parameters used Energy Aware Per-packet Approach Test Simulations

PARAMETER	VALUE
Link Rate	1Gbps
Delay for each Link	100μs
Long flow Size	10Mb
Short flow Size	10Kb
Packet Size	1Kb
Buffer capacity of each Switch	1Mb
Congestion Threshold at each link	75% of link capacity

A fat tree topology with 4 pods provides four shortest paths between any two hosts. We have selected a buffer size equal to 1000 packets, where each packet is 1Kb in size. Totalling switch's buffer capacity to 1Mbytes. The DCN uses point to point links of capacity 1Gbps with a delay of 100μs. The congestion threshold for turning all the links on again is set at 75% of the link capacity at each link i.e. 750Mbytes.

3.7.1.4. Traffic Workloads

Traffic workload patterns used for our evaluation is based upon background traffic observed in data center network scenarios, considering long flows as well as short flows. All the test sets consist of varying traffic load on the scale from 1 to 10 Gbps, where traffic flow and parameters are evaluated at each one Gbps traffic load of change. We have used the variable bit rate traffic at each one Gbps traffic load to evaluate certain performance parameters and bandwidth utilization at each link, instead of using the live traffic. Initially within our test topology SDN controller module takes 40.12 seconds, to setup the whole SR/SDN model scenario including OSPF updates, Hello messages and initial decision of links to be turned off. Whenever, threshold is crossed on any active link and all the links are turned on, it takes 8.95 seconds to reroute the traffic and make decision regarding new set of links to be turned off using link selection algorithm. Although, dynamic live traffic can be used for evaluation of our results but we have kept it out of our research focus due to certain complexity issues.

3.7.1.5. Evaluation Criteria

The objective of performing our test evaluation is to find out the advantages of using segment routing for saving energy within a data center network under an SDN environment and to compare the evaluation results of total energy saved or total number of links turned off using a per-flow based approach and per-packet based approach at different traffic load levels. Also, have evaluated flow completion time of short flows at different traffic loads, demonstrating a significant improvement using per-packet based routing. Out of order packets is a problem introduced using per-packet routing approach, which is evaluated at different traffic load levels and finally deducing the possible areas in which our approach will have significant advantages.

For our results we have shown comparison of per-packet based SR model, which is using our proposed algorithm and SDN architecture with per-flow based SR model. Per-flow based SR model uses similar SR model and link selection algorithm for selecting links to be turned off, but it uses per-flow based ECMP load balancing within multiple links. For calculating the Percentage of energy savings achieved using Equation 1 of Power Consumption Model, we have considered the hardware power consumption parameters of Cisco Nexus 2224TP [21].

The tests are repeated for 15 times at each traffic load level varied from 1Gbps to 10Gbps with an interval of 1Gbps and the results and evaluations are listed below.

3.7.2. Results and Discussion

In this section, we have discussed results from our evaluation tests performed.

3.7.2.1. Flow Completion Time of Short Flows

Figure 3.4, shows the flow completion time for short flows along with the traffic related to long flows within the available links. The graph shows the results for both per-packet as well as per-flow based SR model approaches. A long flow is introduced at each 2 Gb interval of traffic load and flow completion time for short flow is evaluated at each 1Gb interval of traffic load. The initial delay of 10ms is due to the link delay of 100µs within each 1Gbps of the link. Using our approach it can be observed that the flow completion time for short flows increases less severely as compared to per-flow based routing. It is due to the load balancing of long and short flows in per-packet load balancing approach. Thus, per-packet SR model is having a big advantage of keeping the traffic load balanced within available links for a longer time, leading to less frequent congestions.

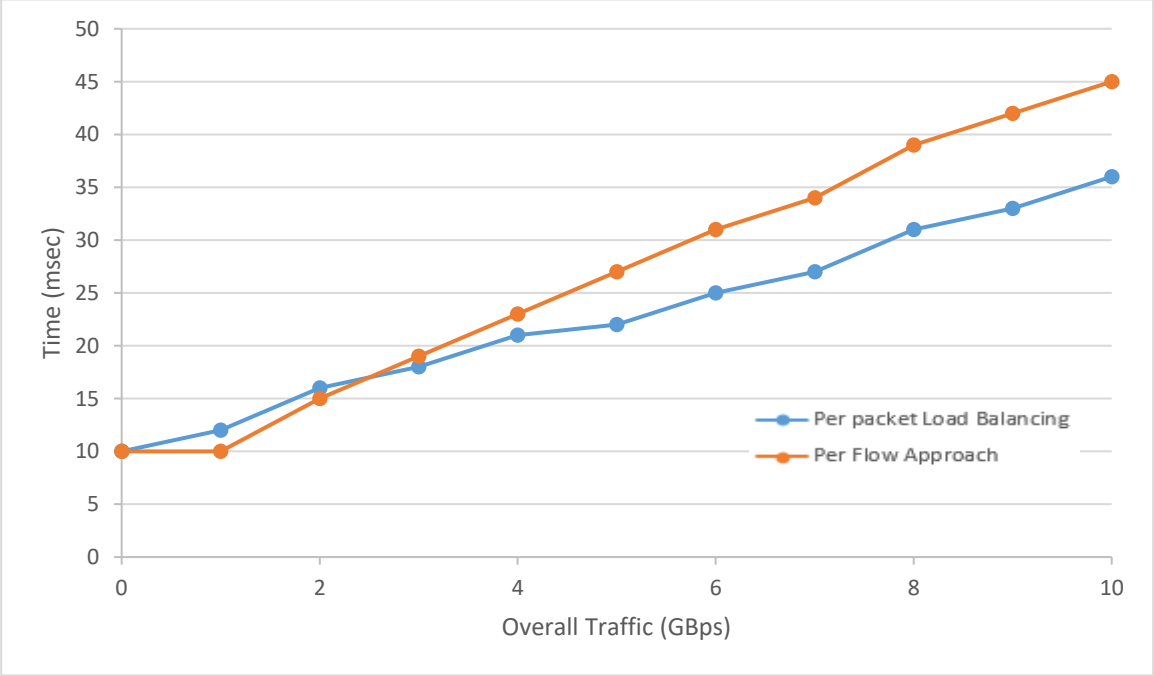


Figure 3.4: Flow Completion Time of Short Flows

3.7.2.2. Percentage of Out of Order Packets

For per-flow based routing there are no out of order packets as the whole flow follows the same path but within the per-packet based routing the packets follow multiple equal cost paths randomly. On an average within a 1Gbps link network with overall traffic load up to 10Gbps, 18% packets are out of order, which can further increase with the increased load on the network. Although out of order packets depend on the delay of a link which further depends on the load on that particular link, so by load balancing the traffic within equal cost multiple paths and increasing the bandwidth of the links within the network, we can achieve further better results. Figure 3.5 captures the out of order packets within our topology with the increasing overall traffic load, using the segment routing based per-packet load balancing as well as per-flow based routing. The margin of Error is ±3% with 95% of Confidence level.

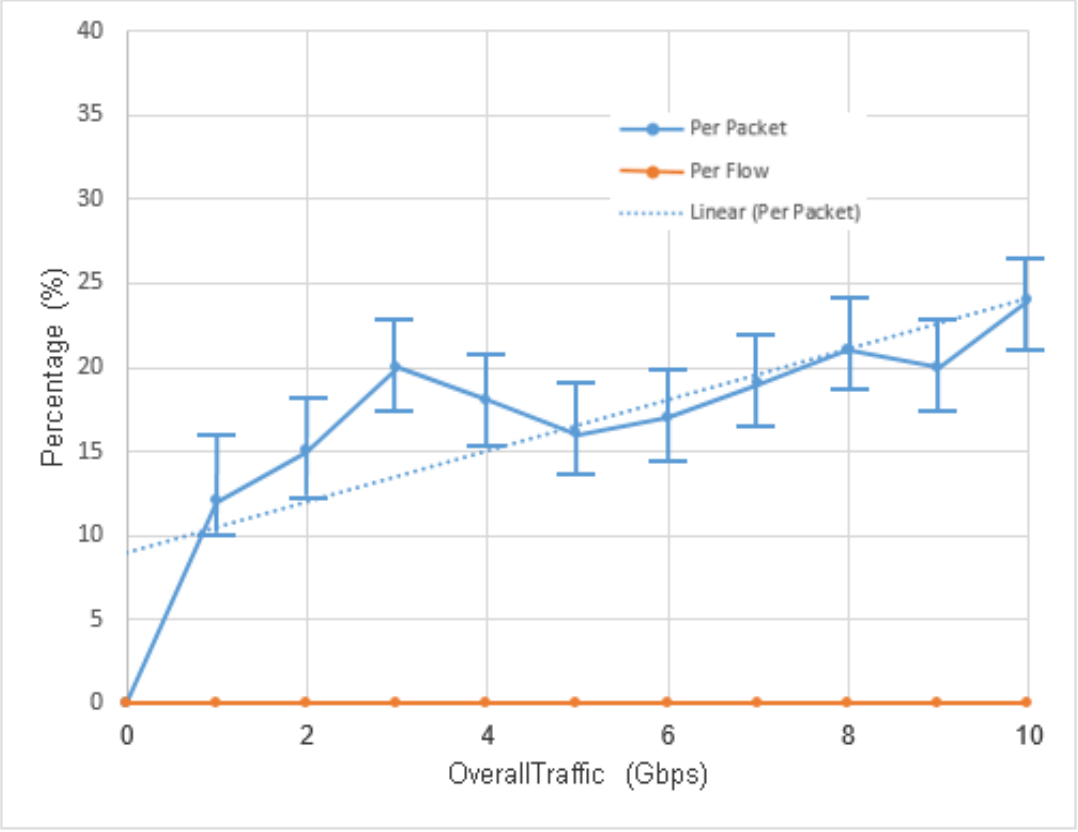


Figure 3.5: Percentage of Out of Order Packets

3.7.2.3. Average Time before All Links are Turned On

Figure 3.6 shows the average time for which the links selected using our proposed algorithm remains in sleep position before they are all turned ON due to congestion within the network, as the traffic load increases a threshold of 75% of link capacity on any active link. The average time before all links are turned ON decreases with the increase in the traffic load, particularly more severe in per-flow based approach. As within per-packet load balancing the flow completion time is better and traffic is load balanced leading to less congestion and more sleeping time for a link within the network. We can also further increase the average sleeping time by increasing the bandwidth of links within the network. But in our simulation, we have used 1Gbps links and observed the sleeping time by varying load within the scale of 1.5 - 6 Gbps. On an average, the margin of error is ±210s with 95% of Confidence level.

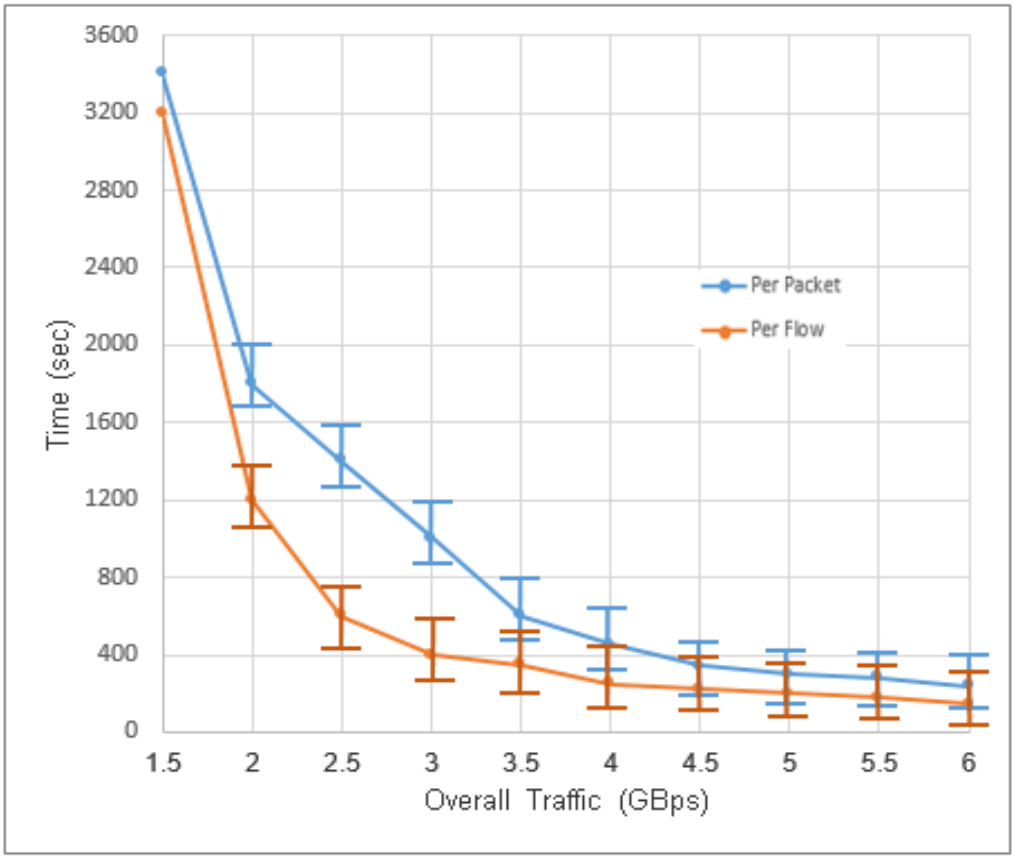


Figure 3.6: Average Time before All Links are Turned On

3.7.2.4. Percentage of Links Turned Off

Figure 3.7, depicts the percentage of links that are turned off using our proposed algorithm within a scale of 0-10 Gbps traffic load. Initially, almost 55% links are turned off when there is no traffic as the SDN controller keeps track of previous traffic load patterns and turn the links off according to that. Percentage of links turned off using per-packet load balancing approach decreases less severely as compared to per-flow approach because the traffic is load balanced within multiple equal cost links which further leads to less congestion and more sleeping period of links. On an average, for per-packet approach, almost 43% of links are turned off for the traffic load till 10Gbps. Overall the number of inactive links decreases as we increase the traffic load, which can be improved by increasing the bandwidth of links within the network to support more traffic. Margin of Error is ±6% with 95% of Confidence level.

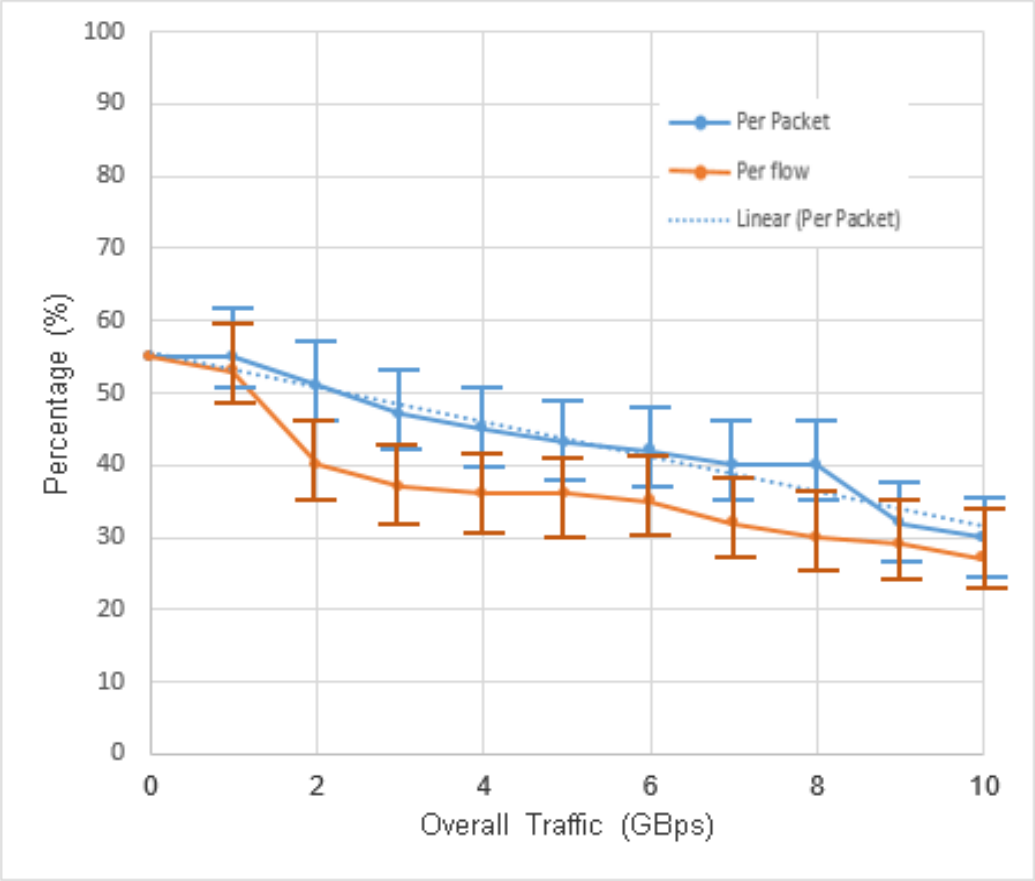


Figure 3.7: Percentage of Links Turned Off

3.7.2.5. Percentage of Energy Saved

Figure 3.8, shows the percentage of total energy saved from per-packet as well as per-flow based approach based upon our above depicted results and evaluations. We have used the power consumption formula i.e. Equation 1, discussed in Section 2.2.3. The energy saved directly depends upon the number of links that are turned off using our algorithm. The graph shows that energy saved decreases with the increase of the traffic load which is more severe in case of per-flow based approach as compared to per-packet and on an average 21% of energy is saved within a network with traffic load varying from 0 to 10 Gbps.

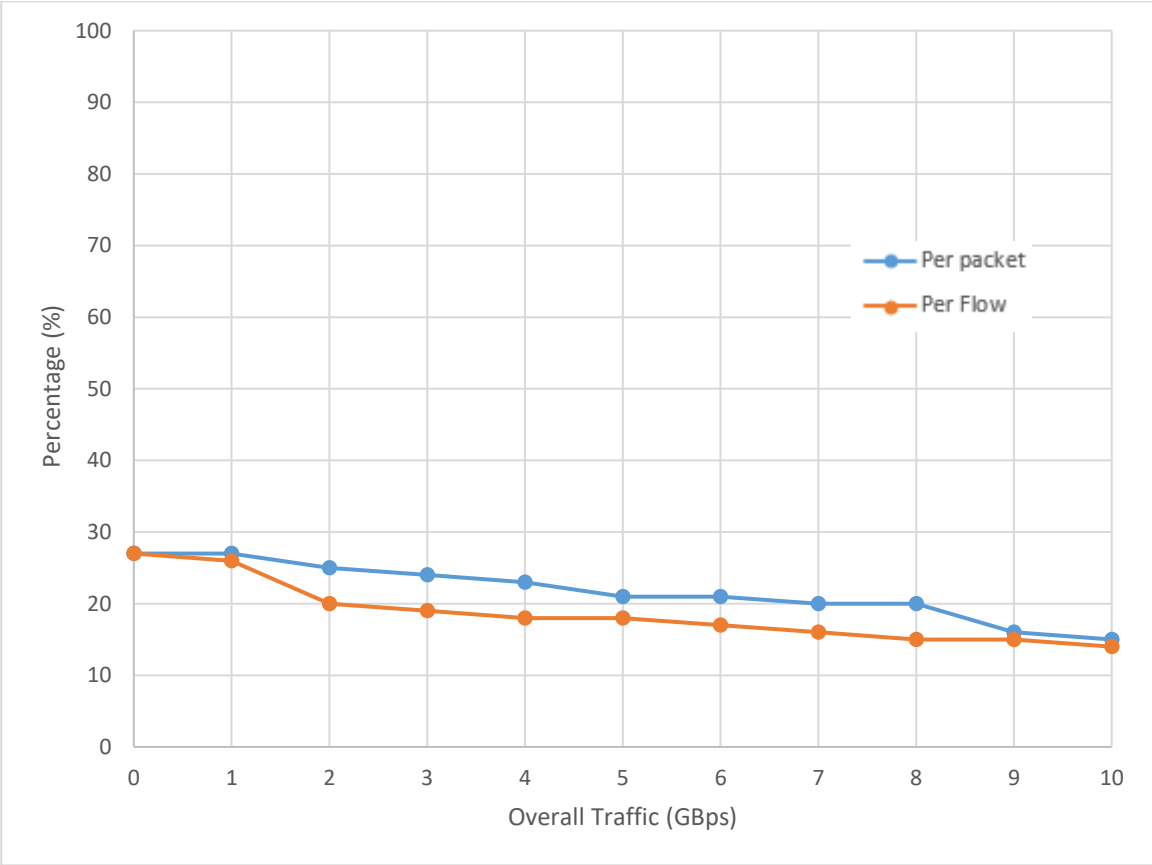


Figure 3.8: Percentage of Energy Saved

3.7.2.6. Result Comparison with STREETE Framework [25]

As we have discussed results in regard to comparison between per-flow and per-packet based Energy efficiency in our proposed SR model. We further strengthen our results by comparing

our approach with the STREETE framework introduced in [25]. While STREETE framework was particularly designed for backbone networks, both the approaches are based upon saving energy using Segment Routing within SDN framework. The link turning off capability of both the approaches is almost same, with the difference lying in managing the traffic more efficiently within available links. Figure 3.9, shows the end to end delay comparison in between per-packet, per-flow and STREETE approach. In between each IEP, the end to end delay was calculated at an interval of 1Gbps of Overall traffic load within the Fat-tree Topology. We have considered 100µs or 0.1ms of propagation delay in all the 1 Gbps links. Due to the processing delay within the per-packet approach and RPS for load balancing the packets between multiple IEP's, per-packet approach incurs more end to end delay as compared to other approaches. Although, the flow completion time for short flows and throughput for long flows is more in per-packet approach as discussed in Section 3.7.2.1. STREETE framework is having lowest end to end delay due to low processing while link selection and it uses minimum hop count shortest path routing, making it efficient in simple topologies like Fat tree topology. Average margin of error for end to end delay for all the approaches is ±0.5ms with 95% of Confidence level.

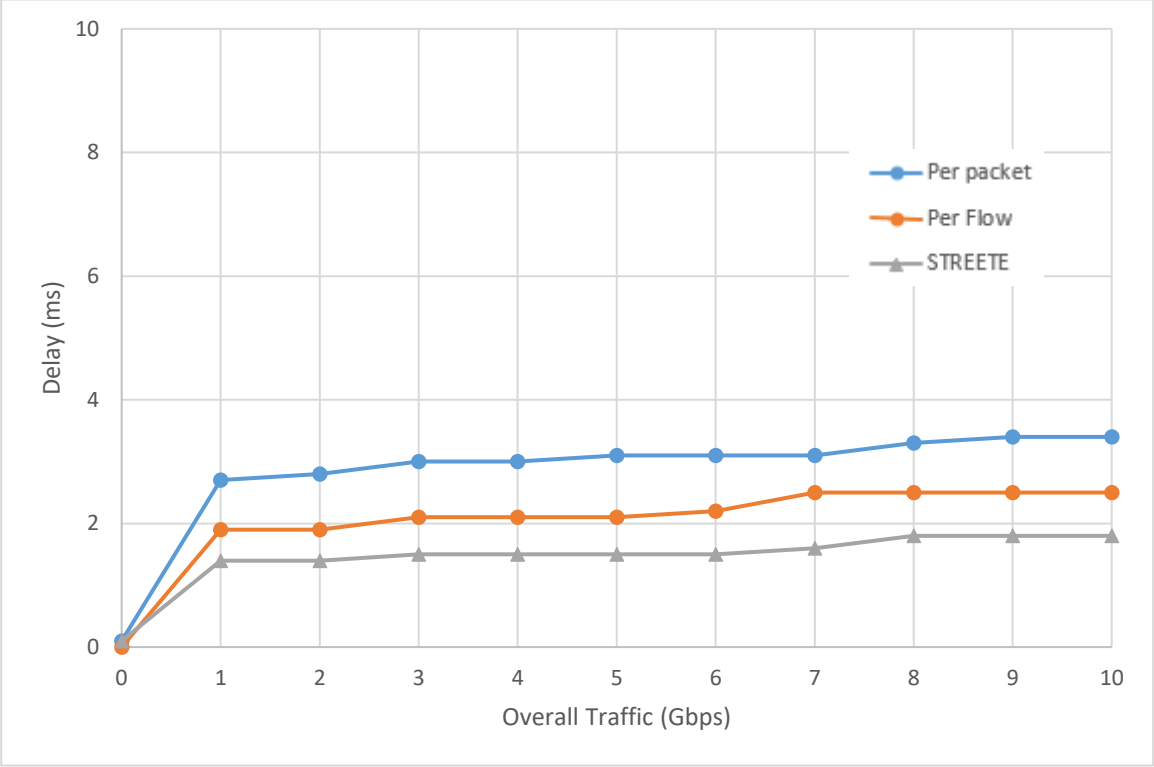


Figure 3.9: End to End Delay

Figure 3.10, shows the average turning sleeping time comparison between per-packet and STREETE architecture approach. As both the approaches consider turning ON all the links, if traffic load on any available link increases more than 75% of the link capacity. Within the STREETE architecture very less emphasis was given on managing the traffic efficiently within available links, leading to frequent congestions and resulting in smaller sleeping period as compared to per-packet approach.

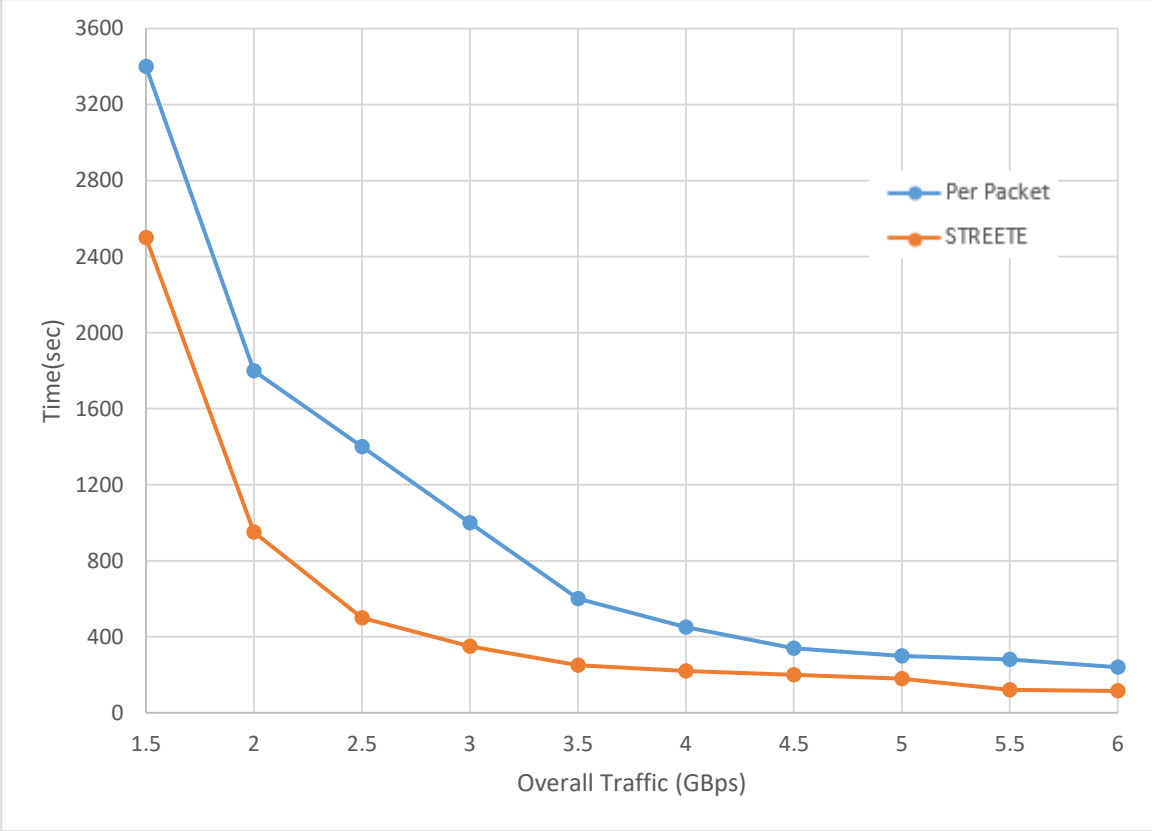


Figure 3.10: Average Time before All Links are Turned On

3.8 Summary

The aim of our research is not only to save energy using SDN based segment routing but also to keep the traffic load well managed using per-packet approach apart from commonly used per-flow based approach. It is evident from our research work, results and their evaluation that segment routing approach leveraging SDN controller advantages can have a significant effect in saving the energy, within the network. Further, we have used per-packet load balancing approach to show its advantages over per-flow based approach in keeping the traffic load well

managed in available links. Results show that per-packet approach has increased the energy savings by keeping links in the sleep state for a longer time, turning off almost 43% of links in our test topology scenario. We also observed that percentage of out of order packets does not have any impact on flow completion time of short flows as well as throughput of long flows in per-packet based approach.

Chapter 4: Per-flow Based Energy and Bandwidth Aware Segment Routing Model for Data Center Networks

In this chapter, we propose a mechanism to save energy within the data center networks as well as effectively utilize the bandwidth using the segment routing based per-flow routing approach. We will be mainly focusing on the flow scheduling method used in the network to improve the bandwidth utilization and further assist in the reduction of energy consumption. In the previous chapter, we have already proposed the mechanism for improving energy efficiency by using per-packet approach, to keep the traffic load balanced among different available links. Bandwidth utilization was not the point of focus in our previous chapter, and the same goes for much other energy efficiency approaches [25-27], which consider only improving the routing methods and turning off as many links as possible for the purpose of reducing energy consumption. As discussed before managing the traffic in a balanced way in available links makes a huge impact in saving energy. But there is still more room for improvement by using efficient flow scheduling method within the network and increasing bandwidth utilization, which helps further in managing the traffic within the available links and increasing the sleeping period of selected links.

SDN provides a set of APIs which simplifies the implementation of various network services including flow management, bandwidth management, routing, and quality of service, using its network-wide view and separated control and data plane [2]. In our approach, SDN controller will help in providing certain perquisites such as bandwidth utilization at each link and size of the incoming flow along with its release and deadline time to implement a flow scheduling approach, which will further assist routing via SR to conserve energy. We will not use per-packet load balancing in this approach, as this chapter is not only oriented towards turning off links but also increasing bandwidth utilization, with the same aim of keeping links in sleep mode as long as possible.

We will be focusing on particularly improving the sleeping time of inactive links by increasing the average time before all links are turned ON, using Link Selection Algorithm proposed in Section 3.4, along with flow scheduling method proposed in this chapter.

Ultimately, improving the energy efficiency within a data center network avoiding the out of order problems introduced by per-packet load balancing approach.

In this chapter, we have described proposed mechanism's design principles in Section 4.1 followed by its architecture in Section 4.2 and its operation in Section 4.3. Simulation and evaluation results are presented in Section 4.5, demonstrating the improvements that are evaluated through simulations using our proposed scheme.

4.1. Design Principles

The motive behind developing this approach is to keep the traffic load managed within the available links using flow scheduling scheme conceptually based on Exclusive routing algorithm (EXR). EXR flow scheduling method, schedules flow in the time dimension and let each flow always exclusively utilize the links of its routing path while transmitting data [53]. Although, there are certain complications within the EXR method such as hard pre-emption due to high priority flows, frequent breaking up of the predefined paths setup for low priority flows and energy wasted for rescheduling and rerouting those flows. Thus, the exclusive routing problem has been converted into flow reservation problem [54], using the knowledge of bandwidth utilization of a link and size, release time and deadline time of flows incoming to the data center network. The flow rate can be shown as follows [54]:

$$\boldsymbol{F_x = \dot{\omega}_x / (D_x - R_x)} \quad (2)$$

Where, F_x = Flow Rate of flow x
$\dot{\omega}_x$ = Flow size of flow x
D_x = Deadline of flow x
R_x = Release time of flow x

Every incoming flow can be converted into a guaranteed flow rate using Equation 2 and flow path will be selected through the links having the minimum bandwidth available to more than the flow rate of that particular flow.

Deployment friendliness: We have deployed our proposed flow scheduling algorithm into the SR-SDN based architecture, which is already proposed in Chapter 3. For further improvement in energy efficiency and bandwidth utilization ratio, we have configured SDN

controller to support flow reservation approach over an SR model. As we have already discussed deployment easiness of segment routing within SDN architecture in Section 3.1, here we will concentrate on deployment specification used regarding flow reservation based flow scheduling approach. The flow reservation algorithm can run as an application over the SDN controller, leveraging the information provided by the controller regarding the state of the incoming flows i.e. their release time, deadline time and bandwidth utilization within each link of the network. Also, as the SDN controller globally manages the state of flows, so the servers do not have to worry about time synchronization amongst each other. The role of SDN controller in further managing the flow states globally will be discussed in Section 4.3.

4.2. Architecture

Our approach is very much based upon the architecture that we have discussed in Section 3.2 of our previous chapter, except we have not used the per-packet load balancing API to configure network elements and controller. The SDN controller is used to program the MPLS data plane to support segment routing and support the flow scheduling method within the network elements. Similarly, IGP's i.e. OSPF-TE or ISIS-TE can be used by the data plane within our architecture to distribute topology information within the whole network, keeping in regard the bandwidth and other administrative constraints, along with an SR configured MPLS data plane to support source routing and forwarding packets based on labels. Whereas, the controller is configured using an API in management plane to manage flow states and perform flow scheduling by maintaining different queues. Also, providing information regarding different network parameters to manage and schedule the flows before their deadline.

- **Flow Scheduling Module:** Flow scheduling API at the top of the controller, within management plane is used to configure the controller to manage the flow states by scheduling traffic flows into different queues i.e. suspended or active queue. Flow scheduling depends upon the available bandwidth over links and available paths within the network. SDN controller then calculates feasible paths and keeps knowledge of bandwidth utilization throughout the network, to schedule the flows. SDN controller

maintains the suspended and active flow queues globally and updates them every time a new flow arrives, or an active flow gets completed.

- **Data Plane Update:** The data plane includes all the functionalities as discussed in data plane module within Section 3.2 of our previous approach. Apart from that, the SR-enabled SDN nodes are required to have the same buffer capacity throughout the network ports. The node controllers attached to the LSRs of the topology is responsible for populating the local forwarding tables, wait for the commands from SDN network controller and informs the controller whenever the status about a particular flow needs to get updated. Different planes within our SDN-based architecture are shown in Figure 4.1, whose functions and operation will be discussed in next section.

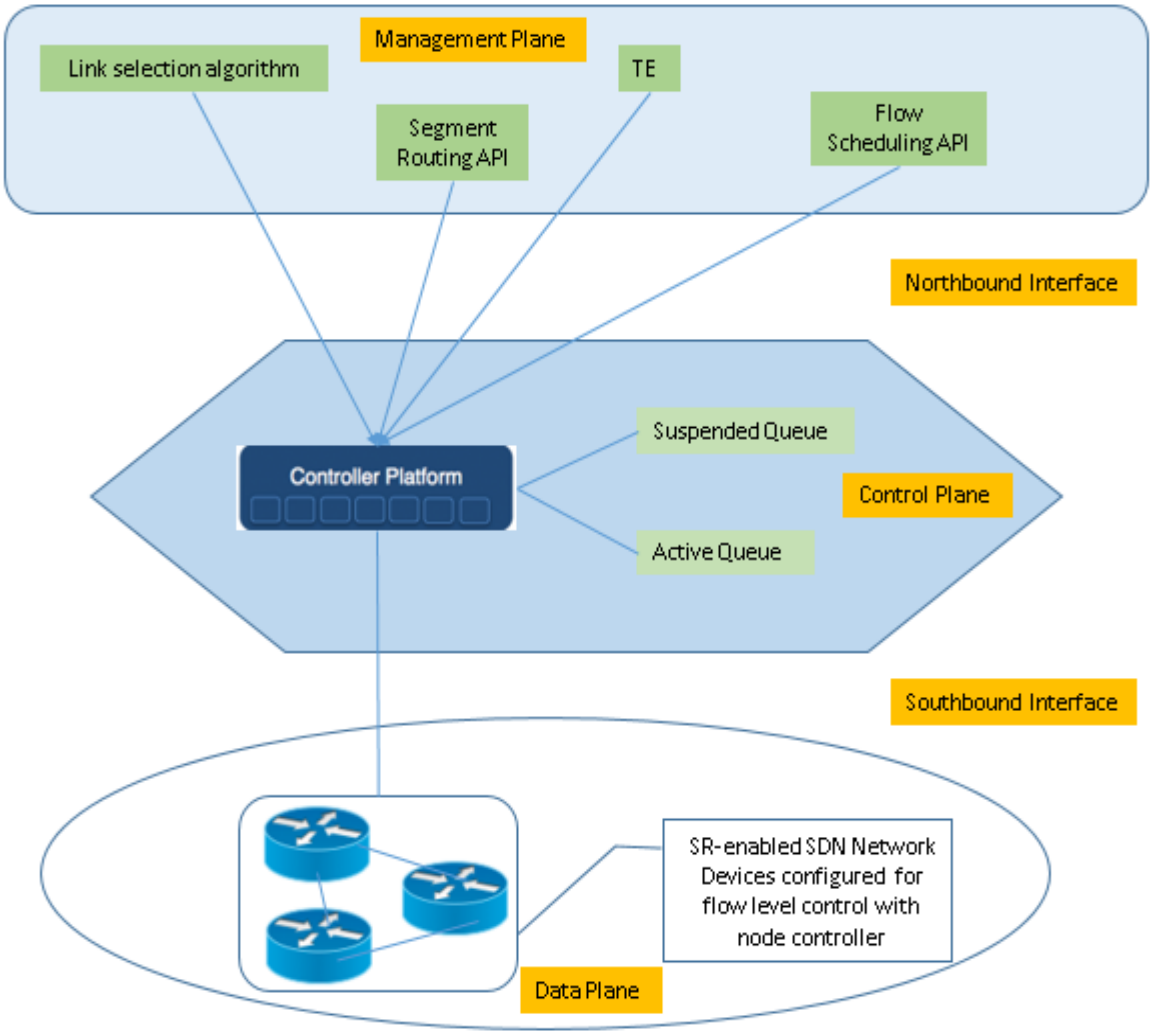


Figure 4.1: Architecture

4.3. Operation

The operation of our flow scheduling approach based SR model regarding deciding which links to be turned off and rerouting paths, is also very much similar to the per-packet based energy efficient SR model, proposed in Chapter 3. The flow scheduling approach is used avoiding per-packet load balancing to efficiently schedule the flows within the available links, after the links are turned off and reroute paths are decided. The flow scheduling helps in efficiently utilizing the bandwidth between the links, as well as managing the flows within the available links in an energy efficient way. Further, avoiding congestion and frequent turning ON of all the links, providing more sleeping time to the turned off links. The whole operation of our approach is divided into four parts, out of which first three are similar to the approach used in Chapter 3. Role and operation of each of the part is discussed below:

1. **Selection of links to be turned off:** The link selection algorithm discussed in Section 3.4, is used in here, to select the links to be turned off. The SDN controller provides the information regarding fulfilling the perquisites like keeping track of traffic matrix and bandwidth utilization throughout the network to implement the link selection algorithm. The link selection algorithm API is used at the top of the control plane to configure the controller and the network elements within the data center network, to gather information and make decisions.
2. **Deciding reroute paths:** The decision for selecting rerouting paths within this approach is also similar to the strategy used in our previous chapter. The SDN controller with its global knowledge and direct programmability link with the node controller of each SR-capable node within the network, makes a decision regarding selecting the best path to reroute the traffic over the set of paths and links that will not be turned off according to link selection algorithm. The use of segment routing paradigm makes it very easy to reroute the traffic, using the SID's over MPLS data plane labels. SDN controller monitors the bandwidth utilization within all the available links, and if any of the links crosses a pre-defined threshold bandwidth usage, all the links are turned on again, and the traffic is routed over to their original shortest path routes.

3. **Actual rerouting & turning links on or off:** The steps followed in turning off the links and rerouting them on the best available paths is the same as that of Figure 3.2, in Section 3.3. Instead of using the per-packet based load balancing, we have used flow scheduling algorithm to manage the flows within the available paths and reroute the traffic according to the decisions made based on output of flow scheduling algorithm.
4. **Flow Scheduling:** The SDN controller is programmed by the Flow Scheduling API to implement the flow scheduling algorithm, by maintaining global active queue and suspended queue within the control plane. Both the suspended and active queue are implemented based upon the priority queue, and within our research work, we have implemented these flows upon the priority based on Early Deadline First (EDF). Both of these queues get updated by the SDN controller, whenever an active flow gets completed, or a new flow arrives. The SDN controller monitors bandwidth utilization at each link and helps to schedule flows according to the bandwidth available in different available paths.

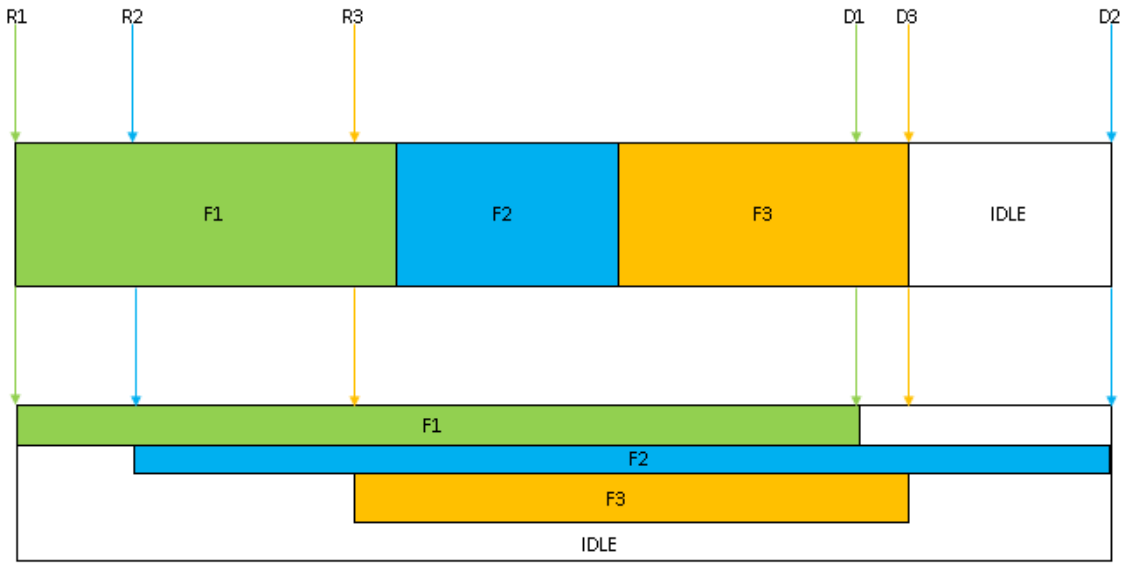


Figure 4.2: Flow Reservation Problem

With the release time R_i, deadline D_i and flow size ω_i, every flow can be scheduled to links with the given time interval using Equation 2. Figure 4.2 shows, three flows F1, F2 and F3 getting converted from exclusive routing problem into a flow reservation problem using

Equation 2, where R1 < R2 < R3 and D2 > D3 > D1. Flow reservation leads to the scheduling of flows into harmonic sets, minimizing energy consumption as well increasing bandwidth utilization within a link, by supporting flows and efficiently executing them parallel within unused bandwidth space. The details about flow scheduling algorithm are discussed in the following section.

4.4 Flow Scheduling Algorithm

The SDN controller selects a particular set of links to be turned off using the Link Selection Algorithm along with rerouting paths. After which, the traffic is rerouted to the available links using the segment routing SID labels and the other links are put into sleep mode using the signaling protocols between the network element and the controller. Then, comes the role of our flow scheduling algorithm to schedule the flows within the available links, utilizing the maximum available bandwidth and managing the flows within available links without congestion. The SDN controller will globally gather the information regarding the set of incoming flows and its parameters like flow size (ὼ), deadline time (D) and release time (I). Also, SDN controller manages the Active flow set priority queue P (a) and suspended queue L(s), updating it every time there is a change in the state of flow or new flow is incoming. The active flow set list is maintained by SDN controller globally and is based upon the working of the priority queue. The flows are given priority based upon their deadline time i.e. Early Deadline First (EDF) and the flow whose deadline is approaching the earliest gets selected. Whenever a new flow arrives, using the information gathered by SDN controller, it will be converted into a flow rate of a particular bandwidth using Equation 2. Now the problem lies in finding a path, which has enough minimum bandwidth to support the flow rate of that particular flow.

Within a data center network, there are more than one paths to reach between an IEP set so that we start with the path with lowest number of hops. If there is a path available, whose links are having minimum available bandwidth more than the flow rate (B) required to reach destination, then the flow is added to P (a) and starts executing according to the priority of that particular flow i.e. its deadline time. Otherwise, if there is no path to support the required flow rate, the flow is added to L(s), maintained by the SDN controller.

Table 4: List of Notations

NOTATION	EXPLANATION
D	Deadline of a flow
R	Release time of a flow
ὼ	Flow size
F (D, R, ὼ)	Set of incoming flows
P (a)	Priority queue with Active flow set
L(s)	Suspended Flow set
$\boldsymbol{B_i}$	Bandwidth required for flow F_i

Algorithm 4.1: Flow Scheduling Algorithm

INPUT: **<** F (D, R, ὼ), P (a), L(s) **>**

if (new flow arrives F_i (D_i , R_i , $ὼ_i$))

// converting into Bandwidth Reservation Problem

$\boldsymbol{B_i = ὼ_i / (D_i - R_i)}$

Find path // path supporting required bandwidth for incoming flow to reach destination

if (path exists)

// add flow into priority queue with active flow set

P (a) = P (a) + F_i

else

// add flow into suspended flow set

L (s) = L (s) + F_i

end if

end if

if (active flow Fi completes)

// completed flow removed from priority queue with active flow set

P (a) = P (a) - F_i

// suspended flows are considered as new incoming flows

F (D, R, ὼ) += L (s)

end if

Whenever an active flow gets completed, it is removed from P (a) and then P (a) is updated by adding all the flows from L(s) into P (a). Suspended flows are now considered as new incoming flows. Their flow rate is again calculated according to their deadline time, and SDN controller looks for a suitable path which can support the new flow rate of suspended flows acting as new flows. We are avoiding pre-emptions by considering flow rate approach according to the deadline, to achieve combinational optimization in the queued flows as pre-emption of the flows will lead to certain difficulties in achieving energy efficiency, by disturbing the path decisions made for other flows by the SDN controller.

4.5. Implementation Details

The whole implementation of our approach is based on the simulation scenario developed for the Chapter 3 framework i.e. within the OMNET++ 5.0 INET framework. Using the very same concept of MPLS module from INET framework and making changes to it, to support segment routing based adjacency and node SIDs over MPLS labels. Using SDN controller with different APIs working over it to configure the controller as well as the network elements, which are used to simulate as well as manipulate the whole scenario according to the developer's requirements. Simple SDN controller with specific functions is used instead of OpenFlow controller to support SR as well as flow scheduling. Flow scheduling API is developed using the OMNET++ and its abstract data type libraries, allowing to make changes to the controller as well as network elements to facilitate the implementation of proposed flow scheduling algorithm. SDN controller with its global coverage and network parameter knowledge, maintains active as well as suspended queues. The data plane of our network is similar to Chapter 3 designed network, where all of the LSRs consist of a node controller instead of a simple OpenFlow switch, to support certain additional functions and communication with the controller. The SDN controller makes a decision regarding the path to be followed by a flow, the rerouting path in case of failure and the scheduling of the flows within the available paths, in a bandwidth as well as energy efficient way.

4.6. Conditions and Constrains

In our simulation, it is assumed that all the links are having the same capacity and the buffer capacity of all the switch ports is assumed same. SDN controller is assumed to have the knowledge of previous traffic matrix patterns to help to make decision regarding turning off, a specific set of links. A pre-defined traffic load threshold is established at each and every link and traffic over the whole network is constantly monitored by the SDN controller. If the traffic load crosses the threshold, all the sleeping links will be turned ON. The active flow-set queue is supposed to work over the functionalities of an abstract data type i.e. priority queue, given the priority according to the early deadline first. Out of order packet problem, which was faced in the previous chapter, will not be a problem in this approach, as we are considering per-flow based routing in this chapter. But we have still kept, turning on of the links one by one, out of scope in this chapter too.

4.7. Evaluation and Analysis

This section includes the details regarding emulation based experimental tests and the criteria used to evaluate our algorithm and approach. This section also covers the results of the evaluation tests and the discussion about the same.

4.7.1. Experimental Setup

Details about the OMNET++ based experimental setup and the INET framework traffic generator used for the evaluation of our approach are incorporated here along with the test topology and the different test parameters and workloads used in our implementation.

4.7.1.1. OMNET++ Based Experimental Network

Regarding the simulation of our flow scheduling algorithm along with the SDN-based network supporting segment routing, we have used OMNET++ 5.0 [52] for emulation and evaluating the results in comparison to our approach proposed in Chapter 3. We have used built-in components based on C++ simulation libraries and framework regarding the creation of virtual topology of different components supporting network simulations. IDE based INET

framework is used to create these modular components, using inbuilt or self-defined libraries; as INET includes the source files and simulation scenarios supporting all wired or wireless simulations. The simulations can be launched directly from the IDE as a normal C/C++ application and perform C++ source-level debugging on it or be launched as a standalone application with dynamic module parameters.

In our network designed for this chapter, the segment routing supporting data plane which is already discussed, developed and used in our previous chapter implementation, is being used to support the SR application within this chapter as well. NED editor and ini files are used to design the modules and change parameters for them. Each and every LSR within the network is using a node controller to support the communication with the network controller, having a network-wide view and fill the forwarding tables, making the network elements smarter and powerful as compared to simple OpenFlow switches. OMNET++ is well suited for performing data center network simulations and to experiment on networks, which are constrained by network properties such as bandwidth, latency, and queueing. For flow scheduling, we have developed an API working on the top of the controller to make changes to the network controller for supporting proposed flow scheduling and managing active and suspended queues in the control layer. Further, the SR-supporting SDN nodes use per-flow based network forwarding, and active flows are scheduled within the network elements according to the priority of early deadline first. Tests were performed using OMNET++ running on Windows desktop with the following configuration - Intel Core i5 with two 2.2 GHz cores and 12 GB of RAM.

4.7.1.2. INET Traffic Generator

For the traffic generation within our simulation, we have used INET TCPBasicClientApp traffic generation module, which is readily available in the OMNET++ INET framework. We can generate constant bit rate or variable bit rate traffic, depending upon the settings of different parameters. In our simulation, we have used the TCP client-server application to generate traffic, and also its parameters can be changed by using different parameters within its ini file. Below are the basic example commands in the ini file:

**.tcpApp [0].typename = "TCPSvr"*

**.tcpApp [0].localPort = 1234*

**.tcpApp [0].thinkTime = 2s*

**.tcpApp [0].requestLength = 100B*

**.tcpApp [0].replyLength = 1MB*

The above commands specify the type of traffic being used, the local port where the traffic will be sent, the time interval between the requests, the length of the request made and the length of the reply, respectively. Although, these parameters can be changed according to the requirements of the developer, by making changes to the ini file of that particular module.

4.7.1.3. Test Topology and Parameter Settings

For simulation and evaluation purposes, we will be using two topologies 4-ary Fat tree topology and B-cube (4, 1) topology within our SDN data plane, using SR-supporting network elements with node controllers. Within the 4-ary Fat tree topology, it includes 3 layers i.e. Core, Aggregation, and Edge. Core consists of four switches, Aggregation and Edge, each consists of eight switches, with a total of 16 client-servers providing multiple paths for each set of IEP pair. Within fat tree topology, each core switch connects to four aggregation switches; each aggregation switch is connected to two edge switches and two core switches, and each edge switch further connects to two servers providing multiple paths. The results regarding the comparison with the approach used in Chapter 3 can be evaluated using the results observed from the simulation of this topology.

The (4, 1) B-cube Topology [55], can be further used to evaluate our approach within different data center networks. The B-cube topology we have used, is a level-1 topology with four B- cubes and 4 number of switches consisting of 4 ports at each switch within each B-cube. In B-cube Topology each of the network element or switch, is a SPRING-LSR, consisting of a node controller module. Further, each switch at level-1 is connected to one server of each B-cube at level- 0. Overall the whole network consists of 16 servers and 8 switches, four at each level. Both the topologies, 4-ary Fat tree topology and (4, 1) B-Cube topology are shown in Figure 4.3 and 4.4, respectively.

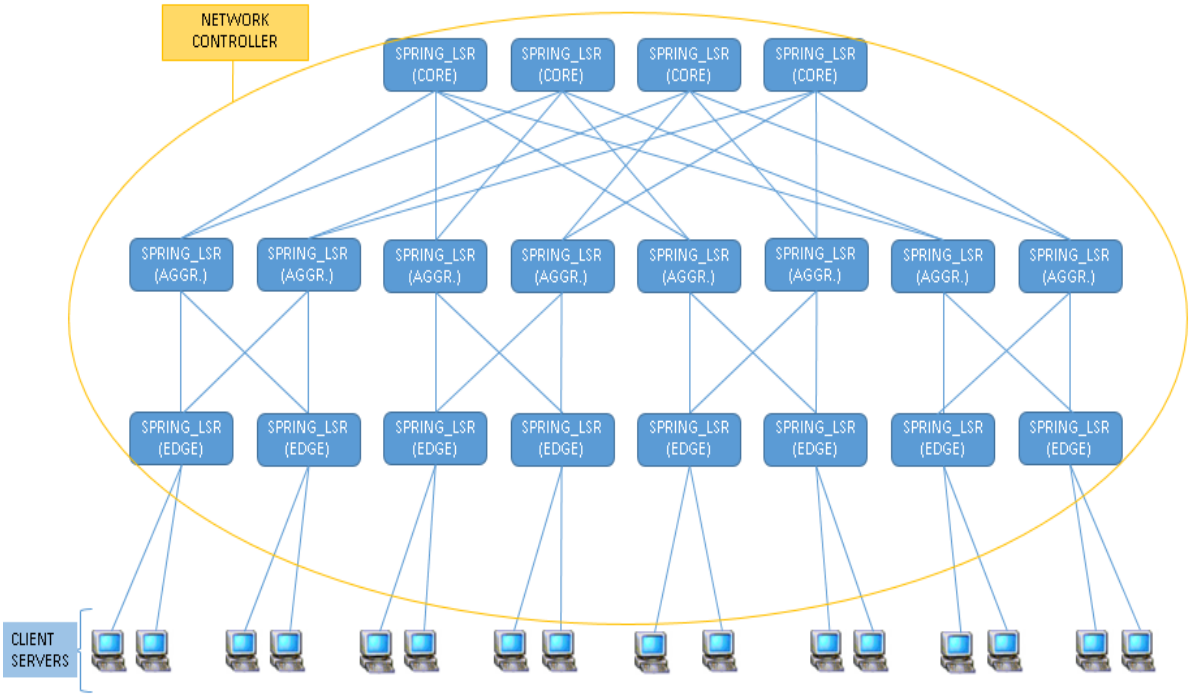


Figure 4.3: 4-ary Fat tree topology

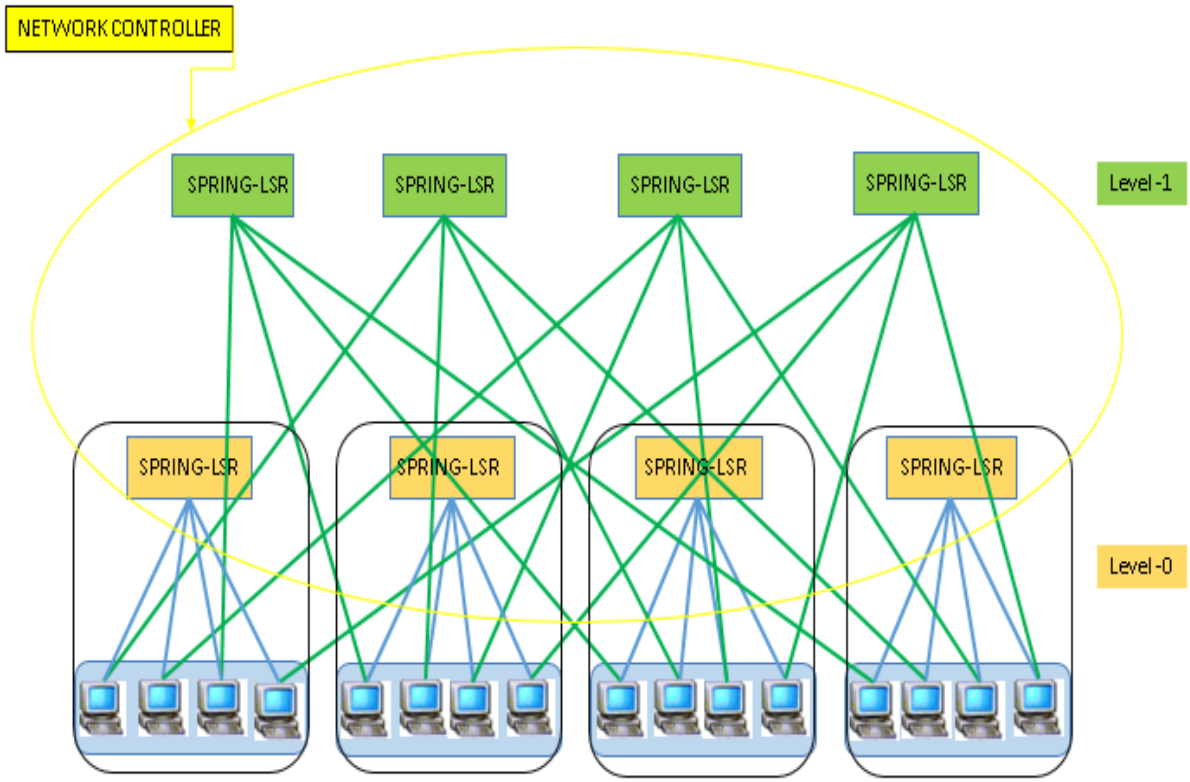


Figure 4.4: (4, 1) B-Cube Topology

Table 5: Parameters used for Bandwidth and Energy Aware Per-flow Approach Test Simulations

PARAMETER	VALUE
Link Rate	1Gbps
Delay for each Link	100µs
Long flow Size	10Mb
Short flow Size	10Kb
Packet Size	1Kb
Buffer capacity of each Switch	1Mb
Congestion Threshold at each link	75% of link capacity

Table 5, specifies the parameter values we have selected particularly for both of our simulation topologies and is similar to the parameter values selected for Chapter 3, to compare between the two approaches proposed. We have selected a buffer size of equal to 1000 packets where each packet is 1Kb in size, totaling switch's buffer capacity to 1Mbytes. The DCN uses point to point links of capacity, 1Gbps, with a delay of 100µs. The congestion threshold for turning all the links on again is set at 75% of the link capacity at each link i.e. 750Mbytes.

4.7.1.4. Traffic Workloads

While evaluating the simulation results within both the data center topologies, the traffic load incoming the network is varied from 1 to 10Gbps by making changes to the ini files, within the traffic generator module of INET framework. At each level of Gbps traffic workload, we have considered a variable bit rate traffic for monitoring bandwidth utilization, flow scheduling as well as parameters to be evaluated in result sets. Initially within (4,1) B-Cube topology, SDN controller module takes 38.65 seconds, to setup the whole SR/SDN model scenario including OSPF updates, Hello messages and initial decision of links to be turned off. Whenever, threshold is crossed on any active link and all the links are turned on, it takes 7.25 seconds to reroute the traffic and make decision regarding new set of links to be turned off using link selection algorithm. Similar to our last approach in Section 3, we have not used

dynamic live traffic within our topologies due to complexity issues, although B-Cube topology has been used to support our approach simulation over different data center networks further.

4.7.1.5. Evaluation Criteria

The evaluation criteria regarding this approach is to show the advantages of using our proposed flow scheduling, in making an SR supporting data center network within an SDN framework, more energy as well as bandwidth efficient. The evaluations regarding the Fat tree topology simulation are to compare with the approach proposed in Chapter 3 and showing the advantages regarding energy efficiency and sleeping time of turned off elements, avoiding the out of order packet problem present in Chapter 3 approach. The B-Cube topology is simulated and evaluated to further show the flexibility of our approach's implementation for different data center networks. The tests are repeated 15 times at each traffic load level varied from 1Gbps to 10Gbps with an interval of 1 Gbps, and the network parameters are evaluated at each 1Gbps interval. For calculating the Percentage of energy savings achieved using our approach and formula 1 of Power Consumption Model we will be considering the hardware power consumption parameters of Cisco Nexus 2224TP [21]. The evaluation results are listed below.

4.7.2. Results and Discussion

In this section, we will be discussing results from our evaluation tests performed.

4.7.2.1. Fat-Tree Topology Evaluation

The evaluation results particularly for fat-tree topology in comparison to evaluation results deduced from the approach used in Chapter 3 are shown below.

4.7.2.1.1. Average Time before All Links are Turned On

As discussed before, it is the average time for which the links to be turned off remains in a sleep mode, until one of the available link's traffic load crosses the threshold of 75% of link capacity. Leading all the links in sleep mode to be turned ON. The graph within Figure 4.5, plots the average sleeping time of links using the per-flow based SR model with flow

scheduling, as well as per-packet based SR model evaluated in Section 3 and compares the result of these two approaches. It can be observed from the graph that average sleeping time for per-flow based SR model with flow reservation is more as compared to the SR model with per-packet based routing. Although the time decreases momentarily at initial traffic load values for both. But there is a small difference, as time decrease is little gradual in the case of bandwidth reservation approach as compared to per-packet based approach, due to better bandwidth efficiency managed by proposed flow scheduling. We can also further increase the average sleeping time by increasing the bandwidth of links within the network, but in our simulation, we have used 1Gbps links and observed the sleeping time by varying load within the scale of 0-10 Gbps.

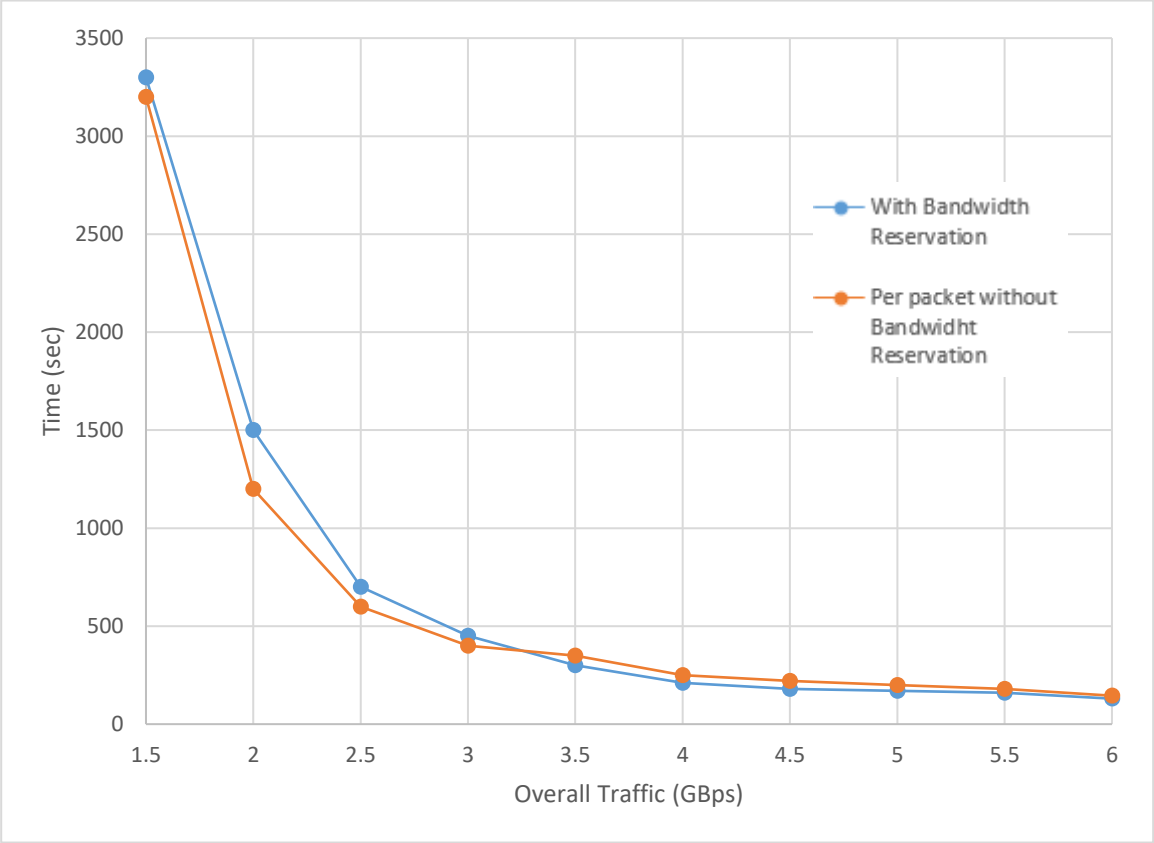


Figure 4.5: Average Time before All Links are Turned On

4.7.2.1.2. Percentage of Links On

Figure 4.6, shows the percentage of links that remain turned ON or active on a varying scale of 0-10 Gbps traffic load, to support the traffic within the network, while the other links are put to sleep using our link selection algorithm. The graph plot within the figure shows the difference of percentage of sleeping links between the two approaches i.e. per-flow based SR model with flow scheduling as well as per-packet based SR model proposed in Chapter 3. Initially similar to the previous approach, 55% of links remains turned off when there is no traffic, as the SDN controller keeps track of previous traffic load patterns and turns the links off according to that. Although there is no significant difference between the outcome of both the approach simulations about the percentage of links turned off, but the change is less severe at low traffic load within the bandwidth reservation approach as compared to per-packet approach. Overall the number of inactive links decreases as we increase the traffic load, which can be further improved by increasing the bandwidth of links within the network, to support more traffic.

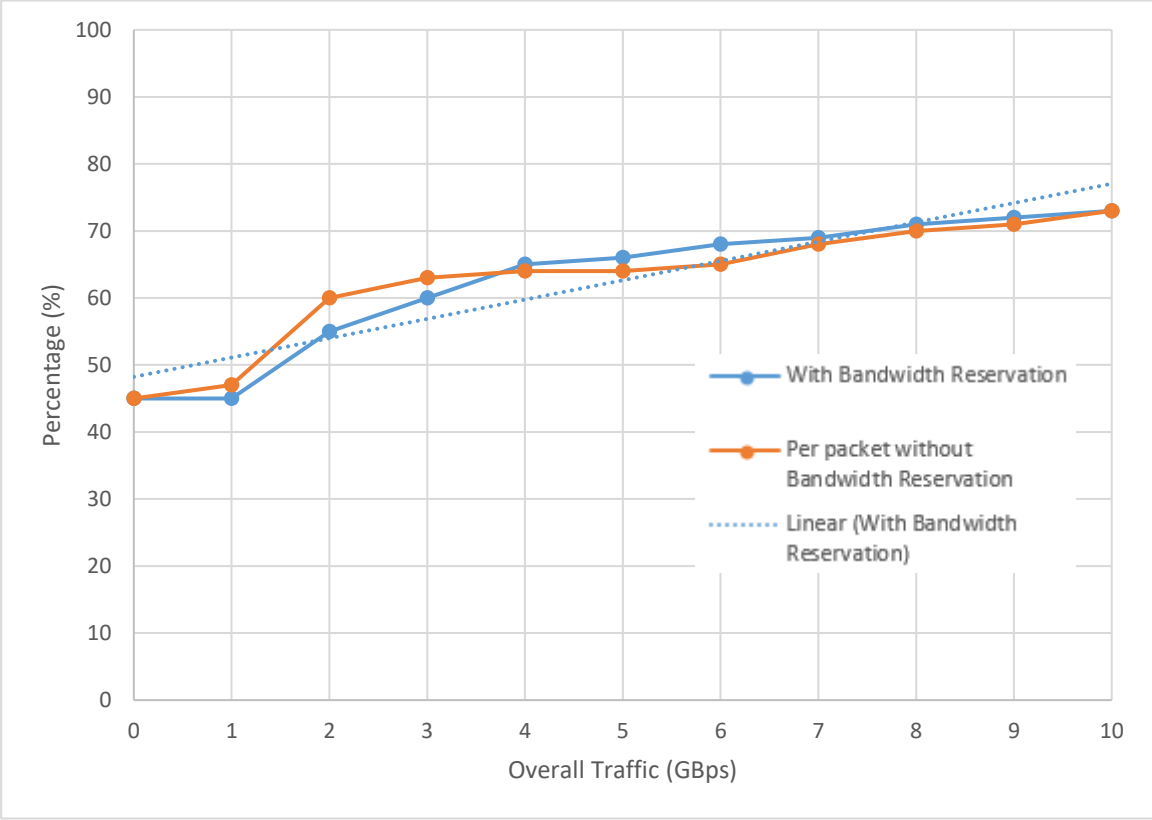


Figure 4.6: Percentage of Links On

4.7.2.1.3. Percentage of Energy Saved

Figure 4.7, depicts the total percentage of energy saved using the SR model with bandwidth reservation along with the per-packet energy saving plot, to assist comparison. We have used the power consumption formula specified in Equation 1 of Section 2.1.3, to measure the percentage of energy saved where the energy saved directly depends upon the number of links that are turned off using our link selection algorithm. Similar to the result set of turned off links for both the approaches, the amount of energy saved by both the approaches are almost equal except the downfall of energy saving is little less severe in the case of SR model with bandwidth reservation. The graph shows that energy saved decreases with the increase of the traffic load.

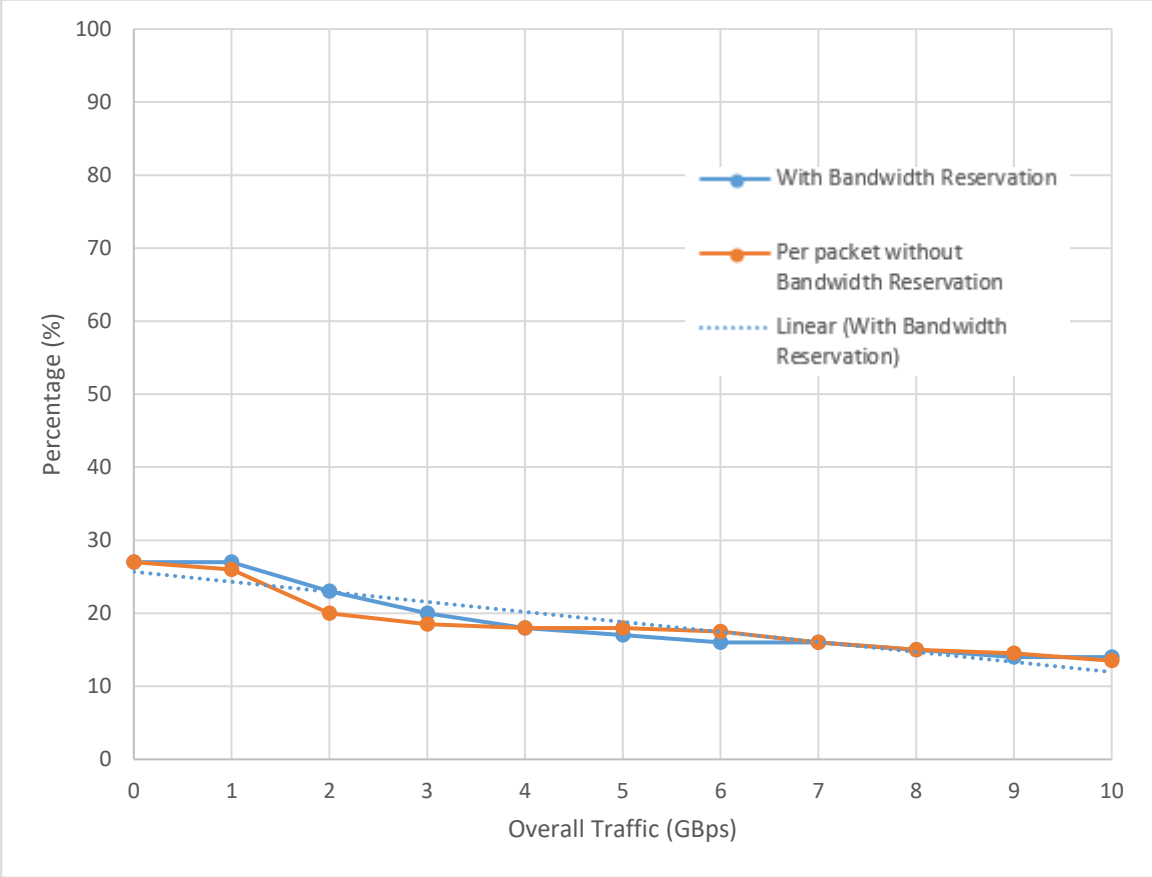


Figure 4.7: Percentage of Energy Saved

4.7.2.2. B-Cube Topology Evaluation

The evaluation results particularly for B-cube topology in comparison to evaluation results deduced from per-flow based SR model approach used in Chapter 3 are shown below.

4.7.2.2.3. Percentage of Links Turned Off

Figure 4.8, plots the percentage of links turned off within a B-cube network with the traffic load varying from 0 to 10 Gbps. Due to the lesser number of multiple path possibilities within a B-cube topology, as compared to the fat tree topology, the ratio of links turned off is lesser in B-cube topology. Although there is not much effect on the sleeping time of the links, as the bandwidth utilization depends on the efficient flow scheduling of flows, incoming the topology using flow reservation. Overall the number of inactive links decreases, as we increase the traffic load, which can be further improved by increasing the bandwidth of links within the network to support more traffic.

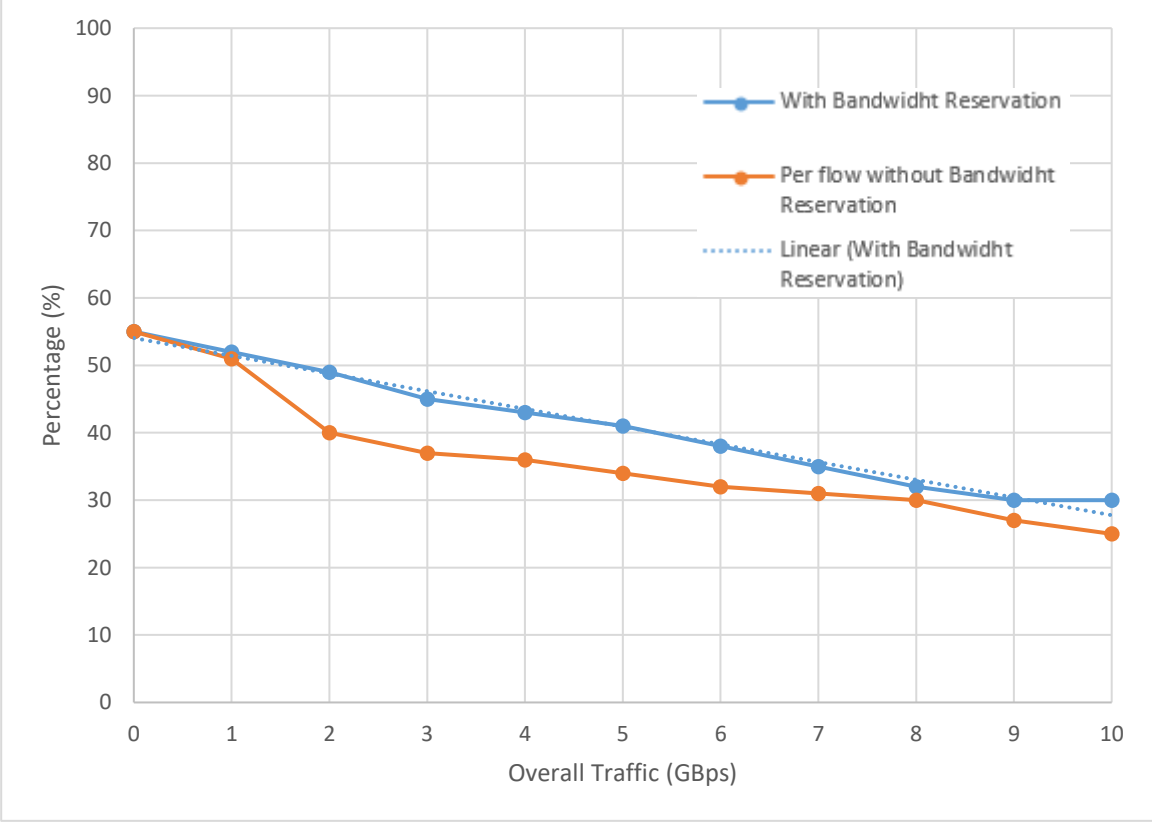


Figure 4.8: Percentage of Links Turned Off

4.7.2.2.2. Percentage of Energy Saved

Figure 4.9, shows the percentage of total energy saved from SR model with flow reservation as well as per-packet based approach, based on results and evaluations deduced within B-Cube topology simulations. We have used the power consumption formula from Equation 1, discussed in Section 2.1.3. The energy saved directly depends upon the number of links that are turned off using our proposed link selection algorithm. The graph plot shows that energy saved decreases with the increase in the traffic load, although the decrease is less rapid in the case of bandwidth reservation based per-flow SR model, which consumes less energy overall due to efficient traffic scheduling.

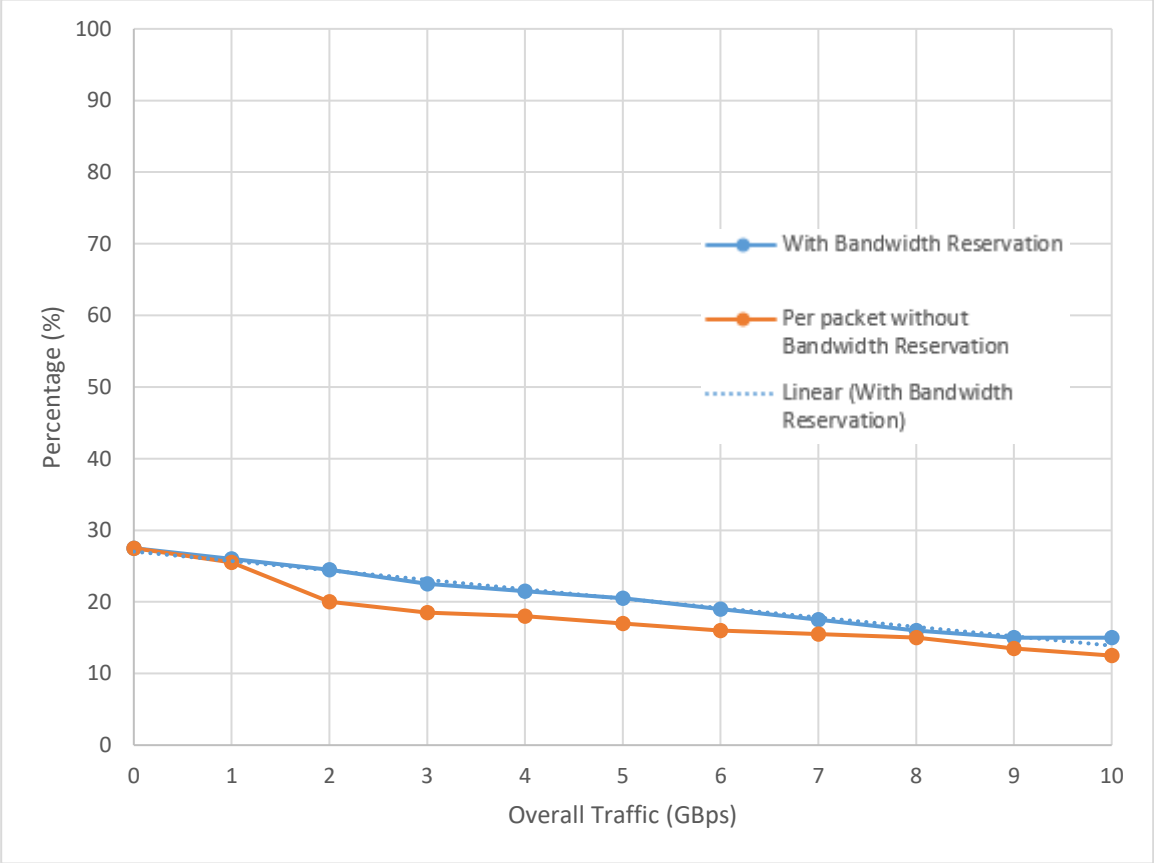


Figure 4.9: Percentage of Energy Saved

3.8. Summary

In this chapter of our research work, we have mostly used the concepts like selecting links to be turned off and the evaluation criteria from Chapter 3. But the main aim of this part was to avoid the out of order packet problem introduced due to per-packet load balancing. Showing the flexibility of our SR based model approach via SDN, even within different per-flow based data center network topologies. It is evident from our results and evaluations from topology simulations that energy efficiency through per-flow based SR model is using almost similar advantages as of per-packet SR model, as discussed in Chapter 3. An upper hand was achieved by avoiding the out of order packet problem and having more bandwidth efficiency due to flow scheduling algorithm used in per-flow SR model via SDN. The results show that the energy saving percentage is almost equal to per-packet SR model approach, with a little difference at the low overall traffic loads. But the energy saving decreases little less severely in Bandwidth Reservation based per-flow SR model as compared to our per-packet approach, as the traffic load increases. B-cube topology is simulated to show the flexibility of our SR-approach with bandwidth reservation, within different data center network topologies and leveraging certain energy efficiency advantages.

CHAPTER 5 - Conclusions and Future Work

In this chapter, we will summarize our research work along with its contribution and conclusion, also outlining the future research work prospects.

5.1. Summary of Work

In our research work, we have described the roundup of Segment routing concept, introduced typically for an SDN environment to benefit the Energy as well as Bandwidth efficiency, within data center network topologies. We have introduced two SR models using per-packet load balancing and per-flow model based on flow reservation, in data center network topologies. In Chapter 2, we have discussed the basic idea behind the introduction of SDN paradigm and its various advantages, along with its usage in the green data center networks for leveraging energy saving benefits. We have also discussed in detail about the SR/SPRING approach introduced by Cisco Systems for operation simplicity and traffic engineering .SR is designed specifically for SDN environment, being the central orientation of our work along with energy efficiency. We have also discussed different research works related to SR, MPLS, and SDN in contribution towards energy efficiency within data center networks.

In Chapter 3, we proposed an SR-model based on per-packet load balancing introduced in SR IETF draft [42] for data center networks, within an SDN architecture. The purpose was to reduce the energy consumption and manage the traffic efficiently within the available links. The architecture of the whole approach is based on SDN paradigm, with an SDN controller centrally managing the data plane. Data plane includes SR-enabled network components along with a node controller, making it more powerful as compared to simple OpenFlow switch. The management plane on the top of controller consists of APIs particularly for SR implementation, per-packet load balancing and link selection algorithm. APIs are used to programme the controller, as well as the network elements for the implementation of our SR-model based on per-packet load balancing. Random packet spraying instead of ECMP is used for load balancing the packets over different equal-cost multiple paths, within a data center network topology used for simulations i.e. 4-ary Fat tree topology. From results and evaluation of simulation, it has been deduced that on an average 43% of links are turned off within a traffic load variation of 1-10 Gbps, where turned off links decreases with the increase of traffic

load. One of our main objectives was to increase the sleeping time of the turned off nodes, which we have successfully achieved by well balancing the long and short flow execution within the available links using per-packet load balancing and simplicity of SR traffic engineering. The results show a significant improvement in flow completion time of short flows and in increasing sleeping time of the turned off flows, as compared to simple per-flow based SR approach using ECMP.

In Chapter 4, we have particularly focused on avoiding the out of order packet problem introduced by per-packet load balancing, making the approach in Chapter3, incompatible with VoIP traffic. Also, it focuses on making the network more energy as well as bandwidth efficient. We have used our proposed SR-model based on per-flow based routing and applied the flow scheduling approach, to increase the bandwidth utilization within the network and converted every flow problem into flow reservation problem using the formula in Equation 2. For implementation, we have used the framework similar to the one used in Chapter 3 along with API for flow scheduling and updated the data plane, to support flow scheduling according to the programming of APIs. Two different data center topologies are used (i) 4-ary Fat tree topology for comparing the results with the per-packet approach and, (ii) B-Cube topology to show our approach's flexibly within different data center networks. The results show that the advantages leveraged from per-flow based SR model with bandwidth reservation, apart from avoiding out of order packets problem, are almost similar to the per-packet SR-model approach. Except the energy saving decreases less severely in per-flow based SR model with bandwidth reservation as compared to the per-packet approach, with the increase of traffic load.

5.2. Conclusion

Our research work demonstrates that the SR approach leveraging the advantages of an SDN architecture, can be used to make the network tasks less complex. Also, the concept of traffic engineering and FRR can be of great simplicity, to be implemented in a data center network at a large scale. By proposing an SR-model via SDN for energy efficiency, we have shown its ease of deployment and compatibility with different paradigms like proposed flow scheduling algorithm and per-packet load balancing approach, using different APIs. A lot of work has

been done earlier on switching the links off in a variety of ways to save energy, but we have not only limited scope of our approach to turning off links but also managing the traffic efficiently within available links using flow reservation and per-packet load balancing. Ultimately, improving bandwidth efficiency along with energy efficiency as evaluated from results.

5.3. Future Work

The approaches used in our research work consider turning on all the links if traffic load on a single link crosses a pre-defined threshold point. Turning On links one by one is avoided in our research work, due to the complexity issues. As it is easy to turn off the links as compared to turning them on because a path selected to be turned on, might be the shortest path for traffic on some other paths. Although, there are few research works which focus on turning links on one by one [56], our approach can be further improved by following this challenging approach leading to more energy saving.

There are a lot of particularly dedicated SDN controller being vastly used nowadays like ONOS [57], OpenDayLight Controller [58], which can have great functionality and scalability advantages in comparison to the OMNET++ used for our simulations. These controllers were not used in our research work because of SR application compatibility and complexity issues along with fewer resources available as compared to OMNET++. The energy, as well as bandwidth efficiency, can further be increased by extending our SR-based approach using these SDN-oriented controllers. Also, the variable traffic being used in our simulation was generated by INET framework of OMNET++; we can further extend our work by using real networks with real traffic matrices.

REFERENCES:

[1] R. Bolla, R. Bruschi, F. Davoli and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," in *Proceedings of IEEE Communication Surveys & Tutorials, vol. 13, issue 2*, pp. 223–244, 2011.

[2] P. Goransson and C. Black, "Software Defined Networks - A Comprehensive approach," *Elsevier*, pp. 352, 2014.

[3] "Segment Routing IETF drafts" [Online]. Available: https://xrdocs.github.io/segment-routing/ietf/ [Accessed: 13-Dec-2016].

[4] P. Mahadevan, P. Sharma, and S. Banerjee, "A Power Benchmarking Framework for Network Devices," in *Proceedings of International Conference on Research in Networking (NETWORKING'09)*, pp. 795-808, 2009.

[5] Greentouch green meter research study, "Reducing the net energy consumption in communications networks by up to 90% by 2020" [Online]. Available: https://s3uswest2.amazonaws.com/belllabsmicrositegreentouch/uploads/documents/GreenTouch_Green_Meter_Research_Study_26_June_2013.pdf [Accessed: 13-Dec-2016].

[6] "CISCO VNI Complete Forecast Highlights" [Online]. Available: http://www.cisco.com/c/dam/m/en_us/solutions/serviceprovider/vniforecasthighlights/pdf/Global_2020_Forecast_Highlights.pdf [Accessed: 13-Dec-2016].

[7] G. Laszewski and L. Wang, "GreenIT Service Level Agreements," in *Grids and Service Oriented Architectures for Service Level Agreements*, pp. 77-88, 2010.

[8] S. Klingert, A. Berl, M. Beck, R. Serban, M. Girolamo, G. Giuliani, H. Meer and A. Salden, "Sustainable energy management in data centers through collaboration" in *Energy Efficient Data Centers, vol. 7396 of the series Lecture Notes in Computer Science,* pp. 13-24, 2012.

[9] D. Meisner, B.T. Gold and T.F. Wenisch, "PowerNap: Eliminating server idle power," in *Proceedings of 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 205-216, 2009.

[10] R. Carrol, S. Balasubramaniam and W.D. Donnelly, "Dynamic optimization solution for green service migration in data centres," in *Proceedings of IEEE International Conference on Communications (ICC),* pp. 1-6, 2011.

[11] D. Sitaram, H.L. Phalachandra, S. Gautham, H.V. Swathi and T.P. Sagar, "Energy Efficient Data Center Management under Availability Constraints," in *Proceedings of Center for Cloud Computing and Big Data Systems Conference (SYSCON), 9th Annual IEEE International*, pp. 377-381, 2015.

[12] X. Meng, V. Pappas and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 29th IEEE Conference on Information Communications (INFOCOM'10),* pp. 1154-1162, 2010.

[13] R. McGeer, P. Mahadevan and S. Banerjee, "On the complexity of power minimization schemes in data center networks," in *Proceedings of the IEEE Global Telecommunications Conference, (GLOBECOM'10),* pp. 1-5, 2010.

[14] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation," *in Proceedings of the 5th USENIX NSDI,* pp. 323-336, 2008.

[15] N. Vasic and D. Kostic, “Energy-aware traffic engineering,” in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pp. 169-178, 2010.

[16] M.R. Celenlioglu, S.B. Goger, and H.A. Mantar. “An SDN-based energy aware routing model for intra-domain networks,” in *22nd International Conference of Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 61-66, 2014.

[17] P. Mahadevan, P. Sharma, S. Banerjee and P. Ranganathan, “Power aware network operations,” in *Proceedings of the 28th IEEE Conference on Information Communications (INFOCOM’09)*, pp. 25-30, 2009.

[18] C. Gunaratne, K. Christensen and S.W. Suen, “Ethernet Adaptive Link Rate (ALR): Analysis of a buffer threshold policy,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM’06)*, pp. 533-534, 2006.

[19] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “Elastic tree: Saving energy in data center networks,” in *Proceedings of the 7th USENIX Symposium on Networked System Design and Implementation (NSDI)*, pp. 249-264, 2010.

[20] R. Tucker, “Will optical replace electronic packet switching,” *SPIE- The International Society of Optical Engineering, Newsroom*, 2007.

[21] “Cisco Nexus 2200 series fabric extenders data sheet” [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-2000-series-fabric-extenders/data_sheet_c78-507093.html [Accessed: 13-Dec-2016].

[22] “IEEE P802.3az Energy Efficient Ethernet Task Force” [Online]. Available: http://www.ieee802.org/3/az/. [Accessed: 13-Dec-2016].

[23] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Elsevier, Computer Networks, vol. 81*, pp. 79-95, 2015.

[24] B. Rodrigues, A. Riekstin, G. Januario, V. Nascimento, T. Carvalho, and C. Meirosu, "GreenSDN: Bringing Energy Efficiency to an SDN Emulation Environment," in *Proceedings of the 14th IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 948-953, 2015.

[25] R. Carpa, O. Gluck and L. Lefevre, "Segment Routing based Traffic Engineering for Energy Efficient Backbone Networks," in *Proceedings of IEEE ICC Advanced Networks and Telecommunications Systems (ANTS)*, pp. 14-17, 2014.

[26] R. Carpa, M.D. Assuncao, O. Gluck and L. Lefevre, "Responsive Algorithms for Handling Load Surges and Switching Links On in Green Networks," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC'16), Green Communications Systems and Networks Symposium*, pp. 73-80, 2016.

[27] M. Lee and J. Sheu, "An efficient routing algorithm based on segment routing in software-defined networking," *Elsevier, Computer Networks, vol.103,* pp.44-55, 2016.

[28] S. Salsano, A. Botta, P. Iovanna, M. Intermite and A. Polidoro, "Traffic engineering with OSPF-TE and RSVPTE: Flooding reduction techniques and evaluation of processing cost," *Elsevier, Computer Communication, vol. 29, issue 11,* pp. 2034-2045, 2006.

[29] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," *Network Working Group, IETF, RFC 3031*, pp. 1-61, 2001.

[30] D. Katz, D. Yeung and K. Kompella, "Traffic Engineering Extensions to OSPF Version 2," *Network Working Group, IETF, RFC 3630,* pp. 1-14, 2003.

[31] H. Smith and T. Li, “IS–IS extensions for Traffic Engineering,” *Network Working Group, IETF, RFC 5305,* pp. 1-17, 2004.

[32] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, “RSVP-TE: Extensions to RSVP for LSP Tunnels,” *Network Working Group, IETF, RFC 3209,* pp.1-61, 2001.

[33] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty and A. Malis, “Constraint-Based LSP Setup using LDP,” *Network Working Group, IETF, RFC 3212*, pp. 1- 42, 2002.

[34] Orhan Ergun, “MPLS Traffic Engineering: Tunnel Setup,” [Online]. Available: http://www.networkcomputing.com/networking/mpls-traffic-engineering-tunnel-setup/442703769. [Accessed: 13-Dec-2016].

[35] M. Zhang, C. Yi, B. Liu and B, Zhang, “GreenTE: power-aware traffic engineering,” in *18th IEEE International Conference on Network Protocols (ICNP),* pp. 21-30, 2010.

[36] B. Addis, A. Capone, G. Carello, L.G. Gianoli, and B. Sansǫ, “A robust optimization approach for energy-aware routing in MPLS networks,” in *Proceedings of International Conference on Computing, Networking and Communications (ICNC’13),* pp. 28-31, 2013.

[37] C. Filsfils, S. Previdi , A. Bashandy, B. Decraene S. Litkowski, M. Horneffer, I. Milojevic, R. Shakir, S. Ytti, W. Henderickx, J. Tantsura and E.Crabbe "Segment Routing Architecture" *draft-filsfils-rtgwg-segment-routing-01, Network Working Group, Internet-Draft,* pp. 1-28, 2016

[38] “*BRKRST-3122- Advanced - Segment Routing: Technology and Use-cases”* [Online]. Available:

https://www.ciscolive.com/online/connect/sessionDetail.ww?SESSION_ID=81907&backBtn=true. [Accessed: 13-Dec-2016].

[39] C. Filsfils, S. Previdi, B. Decraene, M. Horneffer and R. Shakir, "SPRING Problem Statement and Requirements," *draft-ietf-spring-problem-statement-08, Network Working Group, Internet-Draft*, pp.1-16, 2016.

[40] C. Filsfils, S. Previdi, B. Decraene and R. Shakir "Use-cases for Resiliency in SPRING," *draft-ietf-spring-resiliency-use-cases-02, Network Working Group, Internet-Draft,* pp.1-8, 2016.

[41] C. Filsfils, S. Previdi , A. Bashandy, B. Decraene S. Litkowski, M. Horneffer, R. Shakir, J. Tantsura and E.Crabbe "Segment Routing with MPLS data plane," *draft-ietf-spring-segment-routing-mpls-05, Network Working Group, Internet-Draft,* pp.1-15, 2016.

[42] C. Filsfils, C. Previdi, S. , J. Mitchell, E. Aries and P. Lapukhov, "BGP-Prefix Segment in large-scale data centers" *draft-ietf-spring-segment-routing-msdc-02, Network Working Group, Internet-Draft,* pp.1-23, 2016.

[43] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-based segment protection in Ethernet networks," in *Proceedings of IEEE/OSA Journal of Optical Communications and Networking, vol. 5, issue 9*, pp. 1066-1075, 2013.

[44] S. Bidkar, A. Gumaste, P. Ghodasara, S. Hote, A. Kushwaha, G. Patil, S. Sonnis, R. Ambasta, B. Nayak, and P. Agrawal, "Field trial of a software defined network (SDN) using carrier Ethernet and segment routing in a tier-1 provider," in *Proceedings of IEE Global Communications Conference (GLOBECOM) ,* pp.2166-2172, 2014.

[45] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proceedings of IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM),* pp. 205-212, 2014.

[46] L. Davoli, L. Veltri, P.L. Ventre, S. Salsano and G. Siracusano, "Traffic Engineering with Segment Routing: SDN based Architectural Design and Open Source Implementation," in *Proceedings of 4th European Workshop on Software Defined Networking* (EWSDN), pp. 111-112, 2015.

[47] R. Rojas-Cessa, Y. Kaymak, and Z. Dong, "Schemes for fast transmission of flows in Data center networks," in *Proceedings of IEEE Communications Surveys & Tutorials, vol. 17, issue 3,* pp. 1391-1422, 2015.

[48] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe, "Resilient routing using ECMP and MPLS," in *Proceedings of Workshop on High Performance Switching and Routing (HPSR),* pp. 345-349, 2004.

[49] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," in *Proceedings of ACM SIGCOMM Computer Communication Review, vol. 42, issue 4,* pp. 139-150, 2012.

[50] A. Dixit, P. Prakash, Y. Hu, and R. Kompella, "On the impact of packet spraying in data center networks," in *Proceedings to IEEE INFOCOM,* pp. 2130–2138, 2013.

[51] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proceedings to ACM SIGCOMM Computer Communication Review, vol. 41, issue 4,* pp. 266–277, 2011.

[52] OMNET++ with INET: [Online]. Available: https://omnetpp.org/pmwiki/index.php?n=Main.Omnetpp4_ [Accessed: 3-Dec-2016].

[53] D. Li, Y. Shang, and C. Chen, "Software defined green data center network with exclusive routing," in *Proceedings of the 33rd IEEE Conference on Information Communications (INFOCOM'14),* pp. 743-751, 2014.

[54] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu, "Energy-Efficient Flow Scheduling and Routing with Hard Deadlines in Data Center Networks," in *Proceedings of the 34th International Conference on Distributed Computing Systems (ICDCS),* pp. 248-257 ,2014.

[55] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "B-Cube: a high performance, server-centric network architecture for Modular Data centers," in *Proceedings of the ACM SIGCOMM Conference on Data communication,* pp. 63-74, 2009.

[56] ONOS- Open source Network Operating System. [Online]. Available: http://onosproject.org/mission/ [Accessed: 13-Dec-2016].

[57] The OpenDayLight Project: [Online]. Available: https://wiki.opendaylight.org/view/Main_Page. [Accessed: 13-Dec-2016].