T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SCALABLE ROUTING AND RESOURCE MANAGEMENT MODEL FOR SDN-BASED NETWORKS

MAHMUD RASİH ÇELENLİOĞLU
A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER ENGINEERING

GEBZE
2021

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SCALABLE ROUTING AND RESOURCE MANAGEMENT MODEL FOR SDN-BASED NETWORKS

MAHMUD RASİH ÇELENLİOĞLU

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR

PROF. DR. HACI ALİ MANTAR

GEBZE

2021

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TABANLI AĞLARDA ÖLÇEKLENEBİLİR YÖNLENDİRME VE KAYNAK YÖNETİM MODELİ

MAHMUD RASİH ÇELENLİOĞLU
DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
PROF. DR. HACI ALİ MANTAR

GEBZE
2021

GEBZE TEKNİK ÜNİVERSİTESİ	**DOKTORA JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 29/01/2021 tarih ve 2021/05 sayılı kararıyla oluşturulan jüri tarafından 01/02/2021 tarihinde tez savunma sınavı yapılan Mahmud Rasih ÇELENLİOĞLU'nun tez çalışması Bilgisayar Mühendisliği Anabilim Dalında DOKTORA tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : Prof.Dr. Hacı Ali MANTAR

ÜYE : Prof.Dr. Hasari ÇELEBİ

ÜYE : Doç.Dr. M. Kemal ÖZDEMİR

ÜYE : Doç.Dr. Didem GÖZÜPEK

ÜYE : Prof.Dr. Fatih ALAGÖZ

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …..…/…..…/..…… tarih ve ……/…….. sayılı kararı.

İMZA/MÜHÜR

SUMMARY

The volume and diversity of Internet traffic have increased significantly in recent years with the proliferation of mobile devices and networking technologies. Service Providers (SPs) are now striving to make resource management considering recent trends in networking. In this context, Software Defined Networking (SDN) has been alluring the attention of SPs as it provides virtualization, programmability, ease of management, and so on. Yet, severe scalability issues are one of the key challenges of the SDN due to its centralized architecture. First, the SDN controller may become the bottleneck as the number of flows and switches increases. It is because routing and admission control decisions are made per-flow basis by the controller. Second, there is a signaling overhead between the controller and switches since the controller makes decisions on behalf of them.

This thesis proposes an SDN-based scalable routing and energy-aware resource management model (SRRM) for SPs. The proposed model is two-fold. Firstly, it performs routing, admission control, and signaling operations (RASOs) in a scalable manner. Secondly, the controller performs three resource management operations, which are energy-saving, load balancing, route capacity resizing both to save energy and increase bandwidth utilization. To achieve these goals, the model exploits pre-computed routes (PCRs) between each edge pairs in the domain. Experimental results show that the proposed model can successfully perform RASOs in a scalable way, saves energy, and increases link utilization even under heavy traffic loads.

Key Words: Key Words: Software Defined Networking, Resource Management, Energy Saving, Load Balancing, Pre-computed Routes, Scalability, Routing, Optimization

ÖZET

Son yıllarda mobil cihazların ve ağ teknolojilerinin yaygınlaşmasıyla İnternet trafiğinin hacmi ve çeşitliliği önemli ölçüde artmıştır. Servis Sağlayıcılar (SS) ağ yönetimindeki son eğilimleri göz önünde bulundurarak kaynak yönetimi yapmaya çalışmaktadırlar. Bu bağlamda, Yazılım Tanımlı Ağlar (YTA) sanallaştırma, programlama, yönetim kolaylığı vb. özellikleri sağladığı için SS'ın dikkatini çekmektedir. Her ne kadar YTA'ın pek çok avantajı olsa da merkezi mimarisi nedeniyle ciddi ölçeklenebilirlik sorunları vardır. İlk olarak, akışların ve yönlendiricilerin sayısı arttıkça ağ yöneticisi darboğaz haline gelebilir. Bunun nedeni, ağ yöneticisinin yönlendirme ve kabul kontrol kararlarının akış bazında almasıdır. İkinci olarak, ağ yöneticisinin yönlendiriciler adına karar vermesi ve bu kararı anlara iletmesi sebebiyle bu iki ağ elemanı arasında sinyalizasyon yükü oluşmaktadır.

Yukarıda bahsedilen ölçeklenebilir ağ yönetim modeli ihtiyacına istinaden bu tez, SS için YTA ile ölçeklenebilir yönlendirme ve enerjiye duyarlı kaynak yönetimi modeli (SRRM) önermektedir. Önerilen model iki yönlüdür. İlk olarak, önerilen model ölçeklenebilir yönlendirme, kabul denetimi ve sinyalleşme (YKS) işlemlerini gerçekleştirir. İkinci olarak, ağ yöneticisi enerji tasarrufu, yük dengeleme ve kapasite boyutlandırma mekanizmalarıyla hem enerji tüketimini azaltabilir hem de bant genişliği kullanımını artırabilir. Ağ yöneticisi bu operasyonları gerçekleştirmek için, ağdaki giriş-çıkış yönlendirici ikilisi arasında önceden belirlenmiş yollar (ÖKY) kurar. Deneysel sonuçlar, önerilen modelin YKS işlemlerini ölçeklenebilir bir şekilde başarıyla gerçekleştirebildiğini, enerji tasarrufu sağladığını ve yoğun trafik yükleri altında bile verimli kullanımını artırdığını göstermektedir.

Anahtar Kelimeler: Yazılım Tabanlı Ağlar, Ağ Kaynak Yönetimi, Enerji Tasarrufu, Yük Dengeleme, Çoklu Kurulu Yollar, Ölçeklenebilirlik, Yönlendirme, Optimizasyon.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Hacı Ali MANTAR. He has been a great advisor throughout my studies. I will always be grateful to him for giving me this opportunity. Without his guidance and expertise, it would have been impossible to develop the ideas presented in this thesis.

I am also thankful to Fatih TÜYSÜZ, who shared his experiences and assistance through the thesis. All my colleagues in the Department of Computer Engineering at Gebze Technical University also deserve a thank you for their valuable support.

Last but not least, I want to express my greatest gratitude to my beloved wife Şeyma ÇELENLİOĞLU. She was always there for me when I need her assistance of any kind. I also want to express my love to my dear little son Muhammed Ali ÇELENLİOĞLU. Although he did not make my days and nights any easier, he always relieved my stress with his warm smiles. I want to thank Mustafa ÇELENLİOĞLU and Abdurrahman DELİPOYRAZ for their continuous support along this journey. They were interested in my research and helped to expand my vision. I want to thank Semra ÇELENLİOĞLU and Şura DELİPOYRAZ for their unending patience during taking care of me and my family while I was studying. I also thank my all brothers, sisters, and their families. Without their support, affection, and encouragement, I am sure I would not be here right now.

TABLE of CONTENTS

	Page
SUMMARY	v
ÖZET	vi
ACKNOWLEDGEMENTS	vii
TABLE of CONTENTS	viii
LIST of ABBREVIATIONS and ACRONYMS	x
LIST of FIGURES	xii
LIST of TABLES	xiv
1. INTRODUCTION	15
2. SOFTWARE DEFINED NETWORKING	19
3. LITERATURE REVIEW	22
3.1. Control Plane Scalability	22
3.2. Control Plane Scalability and Load Balancing	25
3.3. Energy Efficiency	29
4. SCALABLE ROUTING AND RESOURCE MANAGEMENT	34
4.1. Overview of the Architecture	34
4.2. Virtual Network	36
4.3. Controller Design	38
4.4. Admission Control and Routing	39
4.5. Load Balancing	42
4.6. Route Resizing	46
4.7. Signaling	48
5. ENERGY AWARE RESOURCE MANAGEMENT	50
6. IMPLEMENTATION DETAILS	54
6.1. Databases	54
6.2. Route Management Module	54
6.2.1. Route Establishment	54
6.2.2. Packet Forwarding	56
6.3. Routing and Admission Control Module	57
6.4. Information Collection Module	57

6.5. Load Balancing Module 58
6.6. Route Resizing Module 58
6.7. Energy Saving Module 60
6.8. Inter Controller Communication Module 60
7. EXPERIMENTAL RESULTS 61
7.1. Effect of Route Number on Load Balancing Performance 64
7.2. Effect of Periodic Execution on Load Balancing Performance 65
7.3. Effect of Threshold-based Execution on Load Balancing Performance 66
7.4. Load Balancing Performance under Various Traffic Loads 68
7.5. Load Balancing Performance for Various Topologies 69
7.6. Load Balancing Performance Comparison 71
7.7. Admission Control Performance 72
7.8. Effects of Route Number per Pair and Traffic Load on Energy Saving Performance 74
7.9. Effect of Connectivity on Energy Saving Performance 77
7.10. Comparison of Energy Saving Performance 78
7.11. Comparison of Load Balancing Performance 81
8. CONCLUSION 83

REFERENCES 84
BIOGRAPHY 92
APPENDICES 93

LIST of ABBREVIATIONS and ACRONYMS

Acronyms and Abbreviations		Descriptions
AL	:	Application Layer
CAIDA	:	Center for Applied Internet Data Analysis
CAPEX	:	Capital Expenditure
CL	:	Control Layer
CSS	:	Core SDN Switch
DPID	:	Datapath Identifier
EP	:	Edge Pair
ERMA	:	Energy Aware Route Management Algorithm
ESM	:	Energy Saving Module
ESS	:	Egress SDN Switch
ETSI	:	European Telecommunications Standards Institute
FPTAS	:	Fully Polynomial Time Approximation Scheme
GA	:	Genetic Algorithm
HMA	:	Hash-based Modulo-ρ Assignment Module
ICCM	:	Inter Controller Communication Module
ICM	:	Information Collection Module
ICT	:	Information and Communication Technology
IL	:	Infrastructure Layer
ILP	:	Integer Linear Programming
ISS	:	Ingress SDN Switch
IoT	:	Internet of Things
LAN	:	Local Area Network
LBM	:	Load Balancing Module
LRA	:	Link Rate Adaptation
LSDB	:	Link State Database
MILP	:	Mixed Linear Integer Problem
NDF	:	Network Description File
NSFNET	:	National Science Foundation Network
OPEX	:	Operational Expenditure

PCR	:	Pre-computed Routes
PD	:	Power Down
PDV	:	Pivot Distance Value
PNR	:	Passive Neighbor Ratio
PP	:	Pivot Point
RACM	:	Routing and Admission Control Module
RID	:	Route Identifier
RMM	:	Route Management Module
RRM	:	Route Resizing Module
RSDB	:	Route State Database
RUR	:	Route Utilization Ratio
SDN	:	Software Defined Networking
SP	:	Service Provider
SPN	:	Service Provider Network
SR	:	Segment Routing
SRRM	:	Scalable Routing and Resource Management Model
VLAN	:	Virtual Local Area Network
WAN	:	Wide Area Networks

LIST of FIGURES

Figure No:	Page
2.1: Overview of the SDN Architecture.	19
4.1: Overview of the proposed model.	35
4.2: Illustration of the virtual network obtained from the above network.	36
7.1: Illustration of NSFNET topology.	61
7.2: Illustration of Net-M.	62
7.3: Illustration of Net-L.	62
7.4: Traffic pattern of March 29-30 2015 UTC taken from CAIDA.	63
7.5 :The core link utilizations in time for different number of routes.	64
7.6: The core link utilizations of P2 in time for various load balancing periods.	65
7.7: The core link utilizations of P5 in time for various load balancing periods.	66
7.8: The core link utilizations of P2 in time for various load balancing periods.	67
7.9: The core link utilizations of P2 in time for various load balancing periods.	68
7.10: The core link utilizations for various traffic loads.	69
7.11: Equalization of route costs for NSFNET under moderate traffic load.	70
7.12: Equalization of route costs for Net-M under moderate traffic load.	70
7.13: Equalization of route costs for Net-L under moderate traffic load.	71
7.14: Comparison of load balancing performance of models.	71
7.15: Comparison of admission control time performance of models.	73
7.16: Comparison of flow acceptance rate performance of models.	73
7.17: Active link ratios for P2 under various traffic loads.	74
7.18: Active link ratios for P5 under various traffic loads.	75
7.19: Active switch ratios for P2 under various traffic loads.	75
7.20: Active switch ratios for P5 under various traffic loads.	76
7.21: Active link ratios of 3 topologies having 2 routes per pair under moderate traffic load.	77
7.22: Active switch ratios of 3 topologies having 2 routes per pair under moderate traffic load.	78
7.23: Comparison of energy saving performance based on active link ratio for P2.	79
7.24: Comparison of energy saving performance based on active link ratio for P5.	79

7.25: Comparison of energy saving performance based on active switch ratio for P2. 80
7.26: Comparison of energy saving performance based on active switch ratio for P5. 80
7.27: Comparison of load balancing performance for P2. 82
7.28: Comparison of load balancing performance for P5. 82

LIST of TABLES

Table No:	Page
3.1: Comparison of the SRRM to the Existing Works with Respect to Scalability.	25
3.2: Comparison of SRRM to the Existing Works with respect to Control Plane Scalability and Load Balancing.	29
4.1: Summary of Notations.	37
4.2: Routing Algorithm.	41
4.3: Admission Control Algorithm.	42
4.4: Load Balancing Algorithm.	44
4.5: Load Balancing Algorithm (Contd.).	45
4.6: Load Balancing Algorithm (Contd.).	46
4.7: Route Capacity Resizing Algorithm.	47
4.8: Route Capacity Resizing Algorithm (Contd.).	48
5.1: Possible Energy State Changes of Routes.	52
5.2: Energy Saving Algorithm.	53
6.1: Route-link Matrix of Virtual Network.	59

1. INTRODUCTION

Advancements in both wired and wireless networks, the proliferation of mobile devices and applications have changed the way we use the Internet. Within the last two decades, there are applications and services such as e-commerce, on-line gaming, social media, cloud services, and the Internet of Things (IoT). Accordingly, the volume of Internet traffic has increased, and the Internet traffic characteristic has differentiated dynamically [1]–[3]. Service Providers (SPs) usually serve their network resources such as bandwidth in full capacity to satisfy diverse user demands and accommodate traffic bursts. Network devices such as routers and switches are active all the time, even if network traffic is relatively low compared to peak times [4]. This results in 2-7% of the World's electricity consumption in Information and Communication Technologies (ICTs) [5][6]. The energy consumption of ICTs has been growing by 3% every year [7]. Besides, SPs purchase specialized hardware and software to meet the diverse needs of customers. The management of such networks become complex and costly. Thus, SPs need smart network management systems that not only increase resource utilization but also decrease energy consumption.

Software-Defined Networking (SDN) [8] is a new networking paradigm that emerged to satisfy today's networking demands. It separates control and data planes to provide centralized management, virtualization, agility, programmability, and efficiency. The Infrastructure Layer (data plane) is responsible for performing actions of forwarding and dropping, but not limited to them. Control Layer (control plane) is logically centralized and performs decision-making on behalf of the entities in the data plane. The Application Layer contains applications and services that manage the network. The separation of control and forwarding functions of SDN allows the network control to be programmable, brings intelligence to the network with centralized management, simplifies network design, and allows switch management through open standards. Therefore, the abilities of the SDN allure the attention of SPs. Service providers aim to increase resource utilization, decrease operational costs as well as increase customer satisfaction.

Although SDN has many benefits, there are several scalability problems due to the centralized control plane [9]. First of all, the SDN controller is the only entity that

decides on behalf of all the underlying switches. As the SDN controller makes per-flow basis on-line decisions, it may become a bottleneck in terms of decision-making [10], [11]. Besides, the SDN controller must communicate with all the underlying switches to keep the network state up to date, install new rules, update, or delete the existing rules, etc. This procedure may also result in signaling scalability issue [12]. As a result, using SDN-based network management models in SPNs is challenging due to the scalability problems.

Several works solely address the scalability issues of the control plane in SDN-based SPNs. In these studies, the control plane is distributed, hierarchical, or both. There are also several works that focus on SDN-based resource management. These works either perform traffic load balancing to increase resource utilization or energy-saving both to decrease costs and protect the environment. The works that focus on load balancing distribute the network traffic as fairly as possible among links. Others that focus on energy saving aggregate network traffic to put as many links and switches as possible into sleep mode. Therefore, these two operations are usually considered as opposite. It is still an open issue how to perform routing, admission control, and signaling in a scalable manner as well as energy-aware resource management.

In this dissertation, we design, implement, and evaluate our SDN-based scalable routing and resource management model (SRRM) for SPNs. Our model relies on a central controller and several pre-computed routes (PCRs) between edge nodes. PCRs abstract the complex physical network to a simple virtual one. The virtual network consists of ingress, egress switches, and PCRs. The proposed controller performs routing, admission control, signaling, and resource management operations based on the virtual network. In terms of routing and admission control, the controller does not perform on-line route computation. Instead, it checks the availability of resources and routes the flow to an available route of the corresponding edge pair. In terms of resource management, our model utilizes the trade-off between energy saving and load balancing. To the best of our knowledge, this is the first work in literature that energy-saving and load balancing operations work in harmony. To make this happen, we introduce a trade-off value that determines the importance of these two operations against each other. If trade-off value favors energy-saving, the controller aggregates edge traffic into fewer routes and puts unutilized network elements such as links and switches into sleep mode. If the trade-off value favors load balancing, the controller

saves less energy and focuses on distributing the traffic among routes. In line with the explanations above, we list the contributions of the dissertation below.

- The controller performs routing, admission control, signaling in a scalable manner exploiting pre-computed routes. The controller only takes two related edge switches into account, instead of considering all the switches in the data plane. Also, the controller only installs flow rules to corresponding edge pairs. Therefore, routing, admission control, and signaling become independent from switch size.
- To the best of our knowledge, this is the first work in the literature that exploits the trade-off between load balancing and energy saving. We show that these two mechanisms can work together in harmony. The network administrator adjusts the level of energy-saving concerning utilization. The energy-saving mechanism aggregates the traffic based on this level. The controller puts the unused network elements into sleep mode. Subsequently, it balances the pair load taking active routes into account. Therefore, the controller both saves energy at some level and performs load balancing at the same time.
- We propose a novel adaptive load balancing mechanism. It performs edge pair-based load balancing. This so-called local load balancing, in turn, converges to global (network-wide) load balancing. Besides, our load balancing mechanism takes the current network load into account.
- We propose a novel route capacity resizing mechanism. This mechanism improves the resource utilization further especially in the case of heavy network load.
- The controller can communicate with data plane elements over the modified OpenFlow [13] protocol to put them into sleep and to wake them up.
- The proposed model can be applied to non-SDN-based SPNs by replacing switches on the edge with SDN-capable ones.

The rest of this dissertation is organized as follows. Section 2 describes SDN in details. Section 3 discusses related works. Section 4 presents the scalable routing and resource management mechanisms of the proposed model. Section 5 present the novel energy saving mechanism of the proposed model. Section 6 presents implementation

details of the proposed model. Section 7 presents experimental results and the evaluation. Finally, the work is concluded in Section 8.

2. SOFTWARE DEFINED NETWORKING

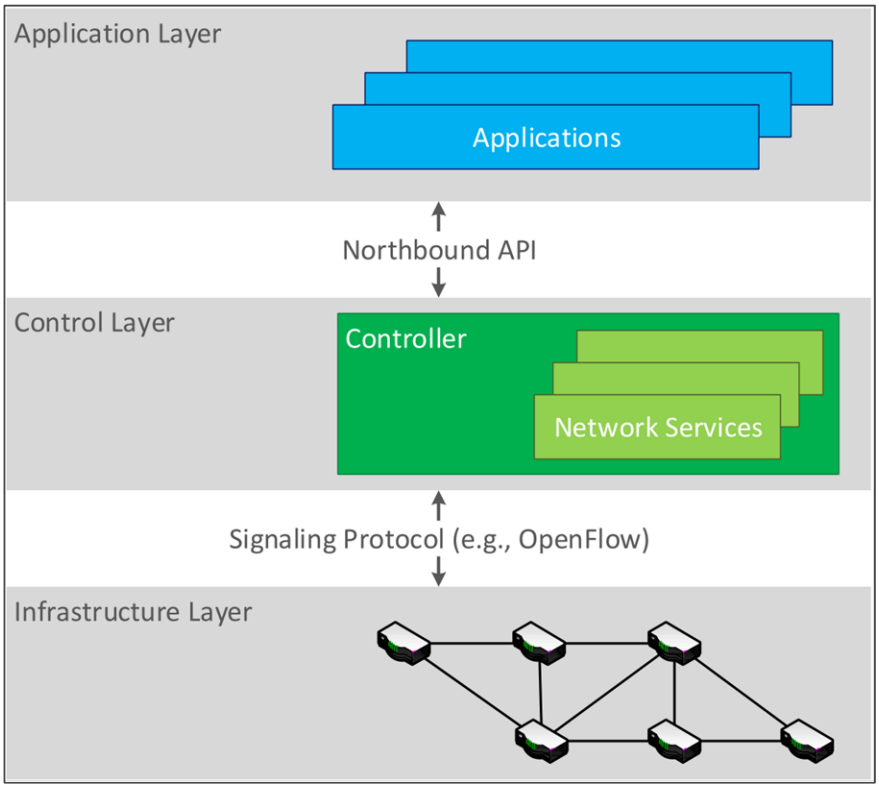

Figure 2.1: Overview of the SDN Architecture.

SDN is the new networking architecture that decouples the layers of traditional networking architecture. The layers of SDN are Infrastructure, Control, and Application as depicted in Figure 2.1. There are also Northbound, Southbound, East/Westbound interfaces.

The Application Layer (AL) is the top layer in the SDN stack. A wide variety of network management applications such as routing, load balancing, firewall, intrusion detection/prevention, policy enforcement, etc. reside in this layer. The northbound interface resides between application and control layer provides an abstraction. Currently, there is no standard for this interface. Some control plane entities (a.k.a. controllers such as ONOS [14], OpenDaylight [15], Floodlight [16]) use REST API for communication with the top layer. Besides, there are also programming languages such as Frenetic [17], Procera [18], Pyretic [19] that allow implementing applications in the top layer. These applications require attention because any kind of bug, security hole, misconfiguration, etc. may cause unintended, and severe operational cost.

The Control Layer (CL) resides in the middle of the SDN stack. It has a global network view and responsible for managing the network. CL serves to the AL via the Northbound interface as mentioned in the previous paragraph. This layer is logically centralized, but it consists of one or more controllers. Thus, it can be both distributed and/or hierarchical. In the distributed control layer, there are multiple controllers each of which manages only a specified part of the network. In the hierarchical control layer, there are at least two levels of controllers, called bottom and top level. Bottom level controllers communicate with both the data layer elements and controllers on the upper layer. Top-level controllers communicate with both the applications on the Application layer and lower-level controllers. The communication between the control and infrastructure layer happens via a signaling protocol such as OpenFlow [13], [20], ForCES [21], and NETCONF [22]. OpenFlow is the de facto signaling protocol in SDN. East/Westbound interfaces allow controllers to communicate with each other. Currently, there is no standard for inter SDN controller communication. Controllers exploit schemes [23], [24], and toolkits [25], [26] for data exchange between each other. There is also an Internet-Draft called SDNi for inter controller communication [27].

The Infrastructure Layer (IL) contains programmable switches. These switches can be both software and hardware. Switches are responsible for the forwarding of the packets as routing functionality is shifted to the control layer. They are also capable of signaling with controllers in the upper layer via southbound interface. An OpenFlow SDN switch has one or more flow tables. A flow table consists of match fields, priority, counters, instructions, timeouts, and a cookie.

- Match Fields: It defines the match rules for incoming packets.
- Priority: It defines the match precedence of the rule among others.
- Counters: Switch updates it if there is a match.
- Instructions: They allow pipeline processing or performing actions on the packet.
- Timeouts: The flow rule expires after the hard or idle timeout. The hard timeout defines the lifetime of the flow rule no matter there are matching packets or not. Idle timeout defines the maximum no match duration of a flow rule.
- Cookie: It is determined by the controller to filter flow statistics, modification, and deletion.

The controller creates a flow entry via sending the OFP_FLOW_MOD OFPFC_ADD message. Upon receiving a packet from its neighbors, SDN switch looks up for a matching rule. If there is a match, the SDN switch performs the defined action in the table. Then, it updates the counters of the rule. Otherwise, the SDN switch either sends the packet to the controller to learn what to do or drops the packet.

Although SDN has many benefits, there are several shortcomings in terms of scalability [9], [28]. Firstly, as SDN switches solely forward flow packets, the SDN controller must decide routing computations and establishment. As the number of flows increases, the centralized control plane may become a bottleneck in terms of decision making especially when there is a single controller. Secondly, the centralized controller must send a request to an SDN switch to receive flow statistics. Thus, the controller may also become a bottleneck when collecting network wide up-to-date information in a large-scale network. Thirdly, the controller must perform per-flow messaging with underlying switches for installing new rules, updating, or deleting the existing rules, etc. This also may result in a control plane scalability issue especially when there are many flows and switches. It is still a research challenge how to perform resource management dynamically in an SPN when there is a single controller.

3. LITERATURE REVIEW

In this section, we review the works that address the control plane scalability issues in terms of decision making and signaling in SDN-based works. We also review the studies that perform load balancing and energy efficiency.

3.1. Control Plane Scalability

The works in [29] and [30] suggest a distributed control plane structure to achieve scalability in the control plane. HyperFlow [29] aims to decrease controller response time through a logically centralized but physically distributed control plane. Controllers are distributed over the data plane. Authors use Publish/Subscribe model for state distribution among controllers. In other words, whenever a state change occurs, the associated controller (publisher) notifies other controllers (subscribers). Onix [30] is a control platform that runs over distributed servers. It provides control logic that allows applications to communicate with network elements. Onix also disseminates network state for consistency among servers. Thus, Onix allows ease of implementation for application developers using Onix API and considering consistency, durability, and scalability trade-offs. Flow-Utility Based Routing (FUBAR) [31] has a centralized QoS routing approach. In this model, traffic is divided into several classes. FUBAR periodically measures the traffic matrix and finds several suitable paths. Then, aggregated flows are divided between corresponding paths to increase resource utilization. The Fibbing [32] approach combines the flexibility and centralized management abilities of SDN with the distributed routing of existing network devices and protocols. Controller fakes routers to direct traffic between paths using faked link-state information. In this approach, the controller also creates backup paths for fast fail-over. ZeroSDN [33] addresses the flexibility of distributed controllers concerning the distribution of control logic. Control logic consists of lightweight control modules that are called controllers and pushes control logic to switches. This way they enable local processing and achieve control plane scalability.

Aside from the distributed control plane approach, works proposed in [34], [35] and [36] provide a hierarchical structure within their control planes to achieve scalability. ASIC [34] adopts a multi-controller approach to solve scalable intra-

domain routing issue in SDN-based large scale networks. The controller in this model has a layered structure. The load balancer in the first layer only divides incoming data flow requests to physically distributed controllers residing in the second layer. Routing is performed in three steps by the chosen controller. First of all, the physical controller gets the data flow initialization request and global network state. Then, it performs routing computations to find a suitable path. In this work, routing, admission control, and signaling are performed in a distributed fashion. Kandoo [35] has two levels of controllers. The first level of controllers handles local events. A controller in this level can control one or more switches, and handle frequent events. On the other hand, a top-level logically centralized controller handles rare events that require a global network view. In [36], the network is divided into zones, each of which has a zone manager. A zone manager is responsible for management operations (e.g., flow setup) within its domain. The network manager is at the top of zone managers, and responsible for network-wide operations such as abstract topology calculation, statistics collection, flow management, credential management for applications and configuration. Unlike previous studies, researchers in [37] propose a network architecture for service providers to reduce controller response time and provide routing scalability. In the proposed architecture, incoming flows are classified into four categories; (i) intra-routing (INR), (ii) MPLS, (iii) QoS, (iv) inter-routing (EXR). Controller pro-actively installs rules for INR and MPLS to improve scalability. Controller steps in only for QoS based routing. The work [38] aims to achieve scalability on multi-domain, multi-vendor SDN-based networks. In this study, there are several domains each of which is controlled by a local controller. There is also a controller at the top called Coordinator Controller, which has a global network view and allows different local controllers to cooperate. Northbound API is unified to achieve this.

There are also works that address the signaling scalability in SDN-based networks. DIFANE [39] is a distributed architecture for enterprise networks. There are two types of switches. Forwarding switches are connected to intermediate switches. If a forwarding switch receives an unknown packet, it redirects the packet to the corresponding intermediate switch. The intermediate switch handles the packet and sends back a response to the forwarding switch to deploy new rules. The controller is responsible to partition rules among switches. Apart from SDN, intermediate switches

also make decisions. Besides, core switches have a heavy load and there is no resource management performed by the controller or intermediate switches. BalanceFlow [40] aims to balance signaling traffic in Wide-Area Networks (WAN) using multiple controllers. In this multi-controller architecture, there is a Super Controller and one or more regular controllers. Super Controller partitions control traffic load among others to provide scalability and low delay. It obtains average number of flow requests per switch and assigns switches to appropriate controllers so as to balance load of controllers. In [41], authors propose a model that addresses to signaling scalability of SDN-based networks. In this model, the controller does not install all flow rules to each switch. Instead of that, it asks switches to propagate the message to the target switch(es). This provides signaling scalability between controller and switches. The model uses proactive flows to handle time-critical flows. Controller installs inactive rules to switches. These rules become active in some cases. Though this model provides signaling scalability, routing scalability is still an issue. The reason is that controller must still perform many routing computations for each flow that does not match with existing rules.

The works mentioned in this part are summarized in the 3.1. In this table, the Ref header indicates the corresponding works. The Control Plane header indicates the control plane arhictecture of the works. The letters S, D, and H stand for single, distributed and hierarchical control planes, respectively. In the single type, there is just one controller in the network. In the distributed type, there are two or more physically distributed controllers in the network. In the hierarchical type, there are layers within the control plane. Each layer has at least one controller. The Network Type header indicates supported networks that are SDN and Hyrid. SDN networks solely contain SDN switches. Hybrid works contain both traditional switches/routers and SDN switches. The Routing header indicates the routing decision types that are online, and offline. In Online type, the SDN controller performs per flow routing computation. In Offline type, the SDN controller does not perform per flow routing computation. Instead, these works usualy exploit static paths. The Signaling header indicates the works that address to the signaling scalability issue of SDN.

Table 3.1: Comparison of the SRRM to the Existing Works with Respect to Scalability.

	Control Plane			**Network Type**		**Routing**		**Signaling**
Ref	**S**	**D**	**H**	**SDN**	**Hybrid**	**Online**	**Offline**	
29		X		X		X		
30		X		X		X		
31		X		X		X		
32		X			X	X		
33		X		X		X		
34		X	X	X		X		X
35		X	X	X		X		
36		X		X		X		
37	X				X	X	X	
38		X		X		X		
39		X		X		X	X	
40		X		X				X
41		X		X		X	X	X
SRRM	X	X	X		X		X	X

As seen in the 3.1, most of the works achieve control plane scalability through multi-controller architecture. Only the work [37] proposes a routing solution for a single controller. In this work, however, the controller performs both online and offline routing. In offline routing, the controller routes flows through pre-established routes for some traffic types such as best-effort. In online routing, controller computes best path for QoS aware flows. Thus, this work is partially scalable in terms of both routing and signaling. In comparison, we perform offline routing per flow. Our work also contains load balancing and energy saving mechanisms.

3.2. Control Plane Scalability and Load Balancing

Some works address both control plane scalability and resource management for SDN-based networks.

Google reports its experience on SDN-based inter-data center resource management (B4) in [42]. In B4, each data center site has a set of controllers. Traffic

engineering server on top of site controllers gathers topological data from them and splits the traffic of source-destination site pair over k-shortest paths. The controller in [43] allocates joint bandwidth for interactive, elastic, and background traffic classes in inter-data center networks. Controller forces proportional fairness within the same traffic class to increase resource utilization.

The authors in [44], [45] propose a load balancing algorithm (A4SDN) derived from the Ant Colony Algorithm. Each packet and switch is treated like an ant and pheromone, respectively. Ants in A4SDN do not converge to a single optimum path, but instead, explore the paths with the weakest pheromone. Similarly, the work [46] also aims to increase resource utilization through load balancing. To do so, the authors modify three different routing algorithms called SWP [47], MIRA [48], and DORA [49] for best-effort traffic. Each of these algorithms performs routing based on estimated available bandwidth in links. MIRA and DORA are suitable for backbone networks as they take the location of ingress and egress switches into account.

Authors in [50] propose a model for hybrid networks where SDN is partially applied to an existing network. In this model, there are SDN and non-SDN switches in the network. SDN switches perform traffic measurement and inform the controller. The controller also collects network information disseminated by OSPF. Subsequently, it leverages this information to change the SDN switch tables for traffic engineering dynamically. To do so, the authors use Fully Polynomial Time Approximation Scheme (FPTAS). Though the system improves resource utilization, the controller performs complex tasks to find optimal routing. Thus, it is not scalable in terms of decision making.

Luo et al. [51] propose a framework (ADMPCF) that computes adaptive multi-paths for centrally controlled networks. ADMPCF uses an existing path as long as the path satisfies the requirements of a flow request. Otherwise, it establishes a new path by computing and assessing several disjoint multi-path finding algorithms in the literature. ADMPCF performs parallel execution of those algorithms to achieve scalability. ADMPCF improves resource utilization through adaptive multi-path establishment based on topology, link cost, application traffic, and network state.

Tuncer et al. [52] propose an SDN-based management and control framework for fixed backbone networks. The proposed framework achieves control plane scalability due to its both hierarchical and distributed architecture. It has local

managers (LMs) at the top. Each LM works with one or more local controllers (LCs). LMs are responsible for implementing the logic of management applications such as resource utilization. LCs perform a sequence of actions based on the configurations given by LMs. The authors also propose an adaptive load balancing mechanism by adjusting traffic split ratios in switches.

The work in [53] aims to achieve both control plane scalability via a distributed control plane and load balancing through switch migration. The proposed work has a set of controllers distributed over switches. A controller is responsible for several switches that reside in the data plane. The proposed work migrates a switch from one controller (source) to another (destination) in the case when the source controller load reaches its maximum capacity. Apart from most of the works in the literature, this study also takes migration efficiency into account and provides a greedy algorithm.

The works [51]–[53] address both control plane scalability and resource management. ADMPCF uses a number of algorithms that work in concert with an adaptive fashion to perform global routing and optimum resource allocation. This makes ADMPCF complex in terms of implementation. Besides, ADMPCF achieves control plane scalability through parallel execution of algorithms used in the framework. In other words, the control plane scalability of ADMPCF depends on the processing power of the system (e.g., cluster of servers). ADMPCF also does not perform admission control, and this may cause over-utilization of resources. Moreover, ADMPCF has partial signaling scalability due to the fact that the controller has to deploy multi-path rules to all the switches along paths if existing paths do not satisfy flow constraints. The framework presented in [52] also has several drawbacks. First of all, the proposed framework requires several controllers and also managers to achieve control scalability due to both hierarchical and distributed control plane architecture. This may increase the CAPEX and OPEX of SPs. Additionally, there appear different communication overheads such as LMO-LCO, LMO-LM, LCO-LC, and LM-LC apart from the controller to switch communication, although controller-switch communication scalability is achieved by reducing the number of switches per controller. Authors also state that communication overhead in management substrate depends on its topology. Moreover, the proposed framework includes a simple communication protocol, yet the details are not provided. Thus, there may appear integration problems in the case of an environment that contains heterogeneous local

controllers and managers. Finally, the proposed framework requires modification of switches for adjusting traffic split ratios based on a hashing scheme. This increases the deployment cost of the proposed framework. In work [53], it is unclear that which decision making entity (e.g., source controller, external server) performs switch migration operation. In the case of a source controller, several controllers may perform migration to a single controller at the same time. This may dramatically increase the load of the destination controller.

The works mentioned in this part are summarized in Table 3.2. In this table, the Ref header indicates the corresponding works. The Control Plane header indicates the control plane architecture of the works. The letters S, D, and H stand for single, distributed, and hierarchical control planes, respectively. In the single type, there is just one controller in the network. In the distributed type, there are two or more physically distributed controllers in the network. In the hierarchical type, there are layers within the control plane. Each layer has at least one controller. The Network Type header indicates supported networks that are SDN and Hybrid. SDN networks solely contain SDN switches. Hybrid works contain both traditional switches/routers and SDN switches. The Routing header indicates the routing decision types that are online, and offline. In the Online type, the SDN controller performs per-flow routing computation. In Offline type, the SDN controller does not perform per-flow routing computation. Instead, these works usually exploit static paths. The Signaling header indicates the works that address the signaling scalability issue of SDN. LB stands for load balancing. It indicates the works that perform load balancing.

All the works in this part perform load balancing with either cost-aware routing or a specially designed mechanism. In cost-aware routing, the controller takes the current network cost into account for a route computation as proposed in [44]–[46]. The volume of assigned traffic may change in time, and the load becomes unbalanced. This also may cause control plane scalability issues because the controller makes a per-flow decision. In specially designed load balancing mechanisms, the controller monitors the network resource utilization and balance the load by alleviating the flow routes. We adopt both mechanisms. Our controller assigns the flow to a path based on its capacity. We also exploit PCR to balance network load. Additionally, most of the works described in this part have the control plane scalability through multi-controller architecture. Only the works [46] and [50] have a single controller. Both perform

online routing so the controller in these works may become the bottleneck in terms of decision making. In contrast, our model can work with one or more controllers. Even in the case of one controller, our work performs routing, admission control, and signaling using PCR. The works [42] and [43] differ from other works in the way that they perform load balancing for SDN-based data center backbone networks. Our model is suitable for SPNs. Finally, traffic splitting in switches offloads the computation overhead of controllers as in [50] but this requires both signaling protocol and switch support. Therefore, this mechanism necessitates modification in SDN architecture.

Table 3.2: Comparison of SRRM to the Existing Works with respect to Control Plane Scalability and Load Balancing.

	Control Plane			**Network Type**		**Routing**		**Signaling**	**LB**
Ref	**S**	**D**	**H**	**SDN**	**Hybrid**	**Online**	**Offline**		
42		X	X	X		X		X	X
43		X		X			X		X
44-45		X		X		X			X
46	X			X		X			X
50	X				X	X			X
51		X		X		X	X		X
52		X	X	X		X			X
53		X		X		X			X
SRRM	X	X	X		X		X	X	X

3.3. Energy Efficiency

There are two main approaches to energy efficiency in SDN-based networks. These are link rate adaptation (LRA) and power-down (PD) approaches. In the first approach, the link rate is adapted according to the traffic load. In the power down approach, ports, line cards, and integrated chassis of routers and switches are either turned off or put into sleep mode. The link rate adaptation approach contributes less energy saving compared to the power down approach [54], [55]. However, the power

down approach causes routing oscillation and delay [56]. There are also some works that exploit both approaches.

The idea in [57] is to reroute flows on existing paths to adjust link loads in a way that LRA-based energy saving is maximized. In the scope of this study, the idea mentioned above is defined as a Mixed Linear Integer Problem (MILP). Then, three greedy algorithms and one genetic algorithm-based heuristic algorithm (GA) are proposed for redirecting flows over existing paths. Experimental results show that GA outperforms greedy algorithms. In this work, several paths are computed per pair before rerouting, whenever the controller executes proposed algorithms. Thus, the control plane and signaling scalability are limited. Although two real networks are used for the assessment of the proposed algorithms, real traffic trace is not used. In addition, there are also no admission control and resource management in terms of load balancing mechanisms.

The following works utilize the PD approach to perform energy saving. The work [58] provides a 0-1 Integer Linear Programming (ILP) model that maximizes energy saving in a global manner by taking the energy consumption of integrated chassis, line cards, and ports of routers into account. It proposes two greedy algorithms namely Alternative Greedy Algorithm (AGA) and Global Greedy Algorithm (GGA) for energy saving. These two algorithms mainly reroute flows on different paths when the network has a relatively low traffic load. This work has routing and signaling scalability issues. The controller performs rerouting per flow. Besides, it communicates with all the switches along the new path per flow.

The work [59] combines IEEE802.az, which is energy-efficient Ethernet and SDN-based Segment Routing (SR) to save energy. SDN controller computes link costs based on two metrics, which are EAGER and CARE. The EAGER metric takes link utilization into account. The CARE metric, on the other hand, takes both link utilization and congestion threshold into account. Two energy-saving algorithms proposed in the work are Tunneling TE (TTE) and Non-tunneling TE (NTE), respectively. In TTE, the controller performs path computation per source and destination pairs in the first phase. Then, the controller establishes SR tunnels. In the final phase, the controller puts unused links into sleep mode. In NTE, the controller computes paths and decides links that will be put into sleep mode. In the second phase, there is no tunnel establishment. The controller only puts the designated links into

sleep mode. Finally, the controller computes ECMP routes between source and destination pairs using awake links. This work in fact has a similar idea that we proposed in our preliminary work [60]. Routing and signaling scalabilities are achieved via pre-determined paths. However, there is no contribution to admission control. Although it is said that load sharing is performed, the authors do not mention how this is achieved.

Software Defined Green Traffic Engineering (SDGTE) [61] framework minimizes active links and switches in backbone networks without knowing the future traffic demand. Authors provide an Integer Linear Programming definition of an energy efficient routing problem. Whenever a flow arrives, the controller performs ILP based routing computation. Apart from energy-efficient routing, SDGTE reroutes flows both in under-utilized and over-utilized links to minimize energy consumption further and to reduce link utilization. In the under-utilization case, links whose utilization is below the predefined threshold are determined, and flows on those links are rerouted. Then, the controller puts these links into sleep mode. In the latter case, the controller identifies the over-utilized links and re-routes the flows on those links to reduce link utilization. The controller performs routing computation per-flow whenever a new flow comes. Besides, the controller also performs re-routing in case of over and under link utilization.

Authors in [62] propose a multi-objective routing approach in a multi-controller control plane. To achieve these objectives, a multi objective evolutionary algorithm, called SPEA2 is developed. SPEA2 performs routing in a way that energy consumption and traffic delay is minimized without degrading the performance. Meanwhile, it takes both controller to switch and switch-to-switch loads into account. Whenever the controller receives a flow request, SPEA2 calculates a path taking data and signaling load into account. Subsequently, the controller establishes a path for the requested flow. The control plane in this work has multiple controllers. Thus, it satisfies routing and signaling scalability at some level.

Green Application Layer is an ETSI standard [63], which is a framework for exchanging information between control and data planes. The work [64] integrates GAL with SDN to exchange information regarding power management of data plane entities with a controller. This model adopts both PD and LRA. In that sense, it proposes an ILP model for the allocation of resources optimally based on network load

and actions of flow tables. The experimental network in this study is small. The controller may become the bottleneck as the size of the network, flows, and actions in flow tables of switches increase. Besides, the signaling scalability issue appears as flows are rerouted.

Authors in [65] aim to minimize active links and adapt discrete link rates to traffic load for saving energy. Firstly, they provide a mixed-integer programming definition of the problem. Secondly, they propose a heuristic algorithm, which identifies most energy-consuming flows and reroutes them on alternative paths for reducing energy consumption. First, the proposed algorithm computes the shortest path for each flow. Subsequently, it calculates the energy consumption of the whole network. The value found at this stage is the upper bound for energy consumption. In the third stage, the proposed algorithm removes each flow one by one to identify its impact on energy consumption. To do so, a weighted graph is generated for each flow. Finally, the controller computes k-shortest paths, and with the least energy consumption is selected as a route. The scheme has routing and signaling scalability issues since the controller performs per-flow routing and rule installation.

Researchers in [66] propose two algorithms for allocation of Virtual SDN (VSDN) in a reliable and energy-efficient manner. Relative Disjoint Path (RDP) generates two trees based on a redundancy factor. Then it merges the two trees to obtain a graph where links and nodes exist in both. In this way, there is only one path if the redundancy factor is 0. Similarly, there are two disjoint paths if the redundancy factor is 1. Otherwise, one or more links are shared among two paths. State-Aware of Bandwidth and Energy Efficiency (SA-BEE) algorithm allocates VSDNs based on available bandwidth and energy consumption factor. It adaptively increases or decreases energy consumption factor based on the network state at first. Then it generates a weighted graph for a source node to all other nodes. Finally, it looks for lower energy consuming paths between the source node and all other nodes.

In GreSDN [67], the controller performs energy-efficient routing without re-routing of flows. It maintains two topologies. The first topology is the physical topology, which contains all the links and switches. The second topology is the virtual topology, which contains only awake links. The controller performs per-flow routing bearing the two topologies in mind. If there is an inactive link along the path, the controller sends a signal to the corresponding switch to awake the link. Meanwhile,

the device management module within the controller periodically checks the state on links. If a link is not used, it puts the link into sleep mode. The authors propose two routing algorithms: Constant Weight Greedy Algorithm (CWGA) and Dynamic Weight Greedy Algorithm (DWGA). Both algorithms perform path computation on two graphs. These graphs are generated per request. At the time of graph generation, the controller removes any link where the summation of the current load with requested bandwidth exceeds its capacity on both physical and virtual graphs. Finally, CWGA and DWGA perform routing computation based on static and dynamic link costs, respectively. The controller performs per-flow routing. Thus, it suffers from a severe routing scalability issue. Besides, the controller must communicate with all the switches along the path during the path establishment process. This also results in the signaling scalability issue.

In the work [68], the authors propose a heuristic algorithm (ETALSA) for energy-saving via only powering up/down links. ETALSA that runs on a controller takes energy prices into account apart from other works. During the execution of ETALSA, the controller iteratively selects switches from the one with the highest energy cost to the lowest. Then, it computes the utility value per link, which is connected to the selected switch. The utility value is computed based on the connectivity of switches, traffic demand, and energy prices. Finally, the controller powers a link down based on the utility value if there becomes less energy consumption. This work resembles our preliminary work [60] in a way that pre-established multi-paths exist both for scalability and resource management. The main differences are that the work [68] powers only the links down and it takes energy prices into account.

Most of the works mentioned in this part solely address the energy-saving in SDN-based networks. Some of the works address both energy and load balancing. Even latter works perform these two operations disjointly. In contrast, to the best of our knowledge, none of the works in the literature performs energy-saving and load balancing in harmony. Additionally, our work proposes how energy-related communication messages can be implemented to OpenFlow. Moreover, our model performs routing, admission control, and signaling in a scalable manner, even there is the only controller.

4. SCALABLE ROUTING AND RESOURCE MANAGEMENT

SPNs usually have over-provisioned bandwidth to satisfy diverse user demands and accommodate traffic bursts. They also have the link and switch/router redundancy for hardware failures. This results in low resource utilization. In such networks, reducing energy consumption and balancing network load among links are available to improve resource utilization. In that sense, the SDN is an emerging network architecture that allures the attention of SPs regarding resilience, virtualization, ease of network resource management, reduction of costs, etc.

Although it has many benefits, a centralized control plane may cause severe scalability issues. Firstly, the SDN controller may become the bottleneck with the increase in the number of flows and switches. Secondly, the SDN controller must take traffic fluctuations into account while optimizing resource utilization since today's network traffic is highly dynamic. Although SDN seems to be adequate for resource management on its own, how to perform scalable routing and resource management in SDN-based SPNs are still open issues.

In this section, we propose a scalable routing and resource management model for SDN-based SPNs. The proposed model is two-fold. On the one hand, the proposed model performs scalable routing, admission control, and signaling in a scalable manner. On the other hand, the controller saves energy, balances load, and resizes route capacities dynamically. To overcome the scalability limitations of SDN, we exploit pre-computed routes to reduce the heavy workload on the controller. Further details of the proposed model are presented in this section.

4.1. Overview of the Architecture

The proposed model is illustrated in Figure 1. As in traditional SPNs, the data plane residing at the bottom consists of provider edge and core nodes. Ingress SDN Switch (ISS), and Egress SDN Switch (ESS) are the nodes where traffic enters and leaves the network, respectively. Any edge node is both ISS and ESS at the same time because traffic is bidirectional. In addition to edge nodes, there are also core switches,

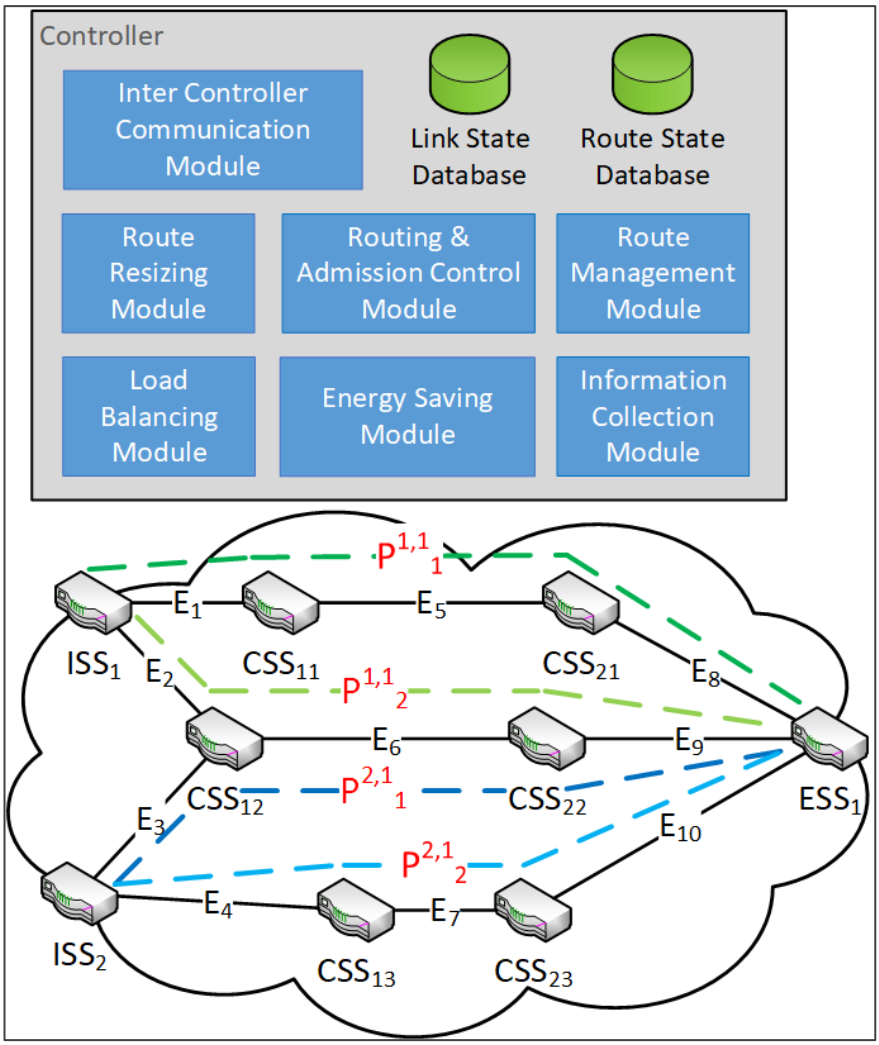

Figure 4.1: Overview of the proposed model.

which we name as Core SDN Switch (CSS). A CSS connects edge switches to each other and solely perform packet forwarding. In the upper layer, the proposed controller resides. It is logically centralized and mainly responsible for resource management (i.e., load balancing, route resizing, and energy saving), routing, and admission control operations. To do so, the controller abstracts the single physical network into multiple virtual networks using PCRs between edge switches. All operations are performed based on these routes. The two layers in Figure 4.1 communicate with each other via a signaling protocol (e.g., OpenFlow).

The proposed model has PCRs between each edge pair (EP), such as ISS1 - ESS1 in Figure 4.1. These routes are virtual and simplify the complex physical network. The simpler virtual network consists of edge nodes and PCRs. For instance, the virtual network presented in Figure 4.1 is depicted in Figure 4.2. The controller performs resource management operations, routing, admission control, and signaling based on

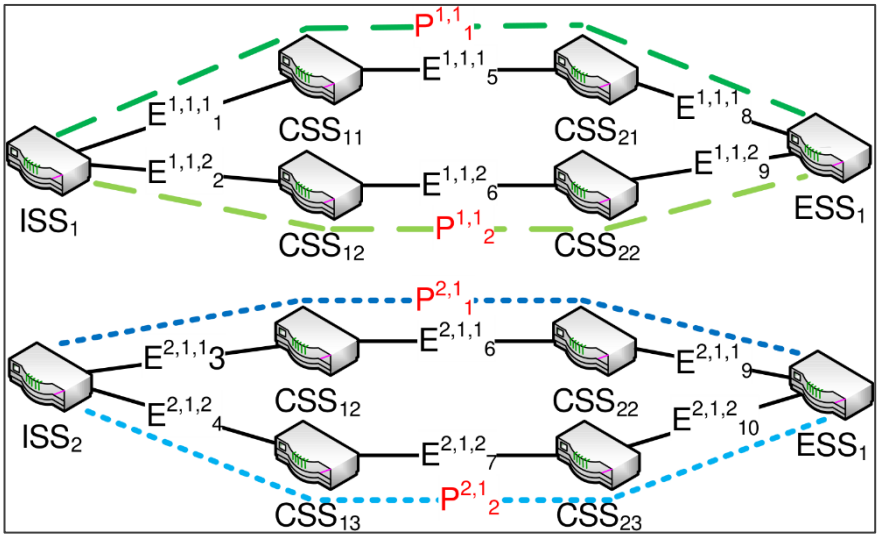

Figure 4.2: Illustration of the virtual network obtained from the above network.

the virtual network. Briefly, the controller aggregates flow into a smaller number of routes adaptive to the traffic load and then puts unused links and switches into sleep mode for energy saving. It also equalizes the cost of routes that belong to the same EP to achieve load balancing. Controller, additionally, adjusts the capacity of PCRs with respect to traffic load to improve resource utilization.

Notice that the controller does not perform routing and admission control operations per flow. In the case of routing, the controller simply assigns a flow to a route. In the case of admission control, the controller decides based on the state of routes, which are stored in both Link State Database and Route State Database. PCRs also allow the controller to manage the whole network with less signaling. This is because the controller does not communicate with each node along the route per flow in routing, admission control, and resource management operations. It only communicates with edge nodes, instead. Therefore, the proposed model achieves scalable routing, admission control, and signaling in favor of PCRs.

4.2. Virtual Network

The physical network is represented by a graph $G = (V, E)$. The set E contains links indexed by i. The set V, on the other hand, contains all nodes (i.e., ISS, ESS, CSS). We denote an ISS-ESS pair as (V_s, V_d) where $V_s \neq V_d$. Note that pair (V_s, V_d) is

Table 4.1: Summary of Notations.

Notation	Meaning
V	Set of switches
E	Set of links
$P^{s,d}$	Set of routes between edge pair (V_s, V_d).
$P_k^{s,d}$	A route belongs to $P^{s,d}$
E_i	A single physical link i
$L_i^{s,d,k}$	Virtual link of $P_k^{s,d}$ on physical link E_i
$\varphi(\dots)$	Capacity of pair, route or link
$\vartheta(\dots)$	Load of pair, route or link
$\lambda(\dots)$	Cost of pair, route or link
$\varepsilon(\dots)$	Energy of pair, route or link

actually the same pair in the reverse direction as (V_d, V_s) since any edge node is both ISS and ESS at the same time. The proposed controller converts the physical network to a virtual one. Then, it performs operations based on the virtual network. The virtual network consists of set $E^{s,d}$ and $P^{s,d}$ of virtual links and PCRs per (V_s, V_d), respectively. We denote a route k that belongs to a pair (V_s, V_d) as $P_k^{s,d}$. Any two different routes $P_k^{s,d}$ and $P_l^{s,d}$ of a pair (V_s, V_d) does not share a link. Thus, they are mutually disjoint. The cardinality of routes per pair ($|P^{s,d}|$) must be at least 2 due to multi-route and can be as many as network topology allows. Besides, the number of routes for each pair can be different. The network administrator makes the decision of which pair has how many PCRs. In the scope of this work, we assume that PCRs already exist in the network. Routes between edge pairs can be established using Virtual LAN (VLAN) [69], MPLS [70], or segment routing [71]. In compliance with this, we do not develop any multi-route computation algorithm as there are several algorithms in the literature such as [72], [51], [73], and [74] that address this subject. $|P^{s,d}|$ per (V_s, V_d) is defined by a network administrator. Note that PCRs can be established regarding any cost value such as residual bandwidth, latency.

Notice that any two routes, each of which belongs to a different pair may share one or more physical links. We name such routes and pairs as neighbor routes and pairs, respectively. For the sake of simplicity, the controller virtualizes links based on

the number of neighbor routes per link. We denote a virtual link that serves a route $P_k^{s,d}$ as $L_i^{s,d,k}$. The capacity of a physical link, denoted as $\varphi(E_i)$, is portioned out to its virtual links based on the load of neighbor routes. In compliance with this, a route capacity $\varphi\left(P_k^{s,d}\right)$ becomes the capacity of its virtual link which has the minimum capacity $\left(\min\left(\varphi\left(P_i^{s,d,k}\right)\right)\right)$. For example, there are two routes on the physical link E_6 in Figure 4.1. Hence, its virtual links are $E_6^{1,1,2}$ and $E_6^{2,1,1}$, where $E_6^{1,1,2}$ serves the route $P_2^{1,1}$ of pair ISS1 - ESS1 and $E_6^{2,1,1}$ serves the route $P_1^{2,1}$ of pair ISS2 - ESS1.

4.3. Controller Design

The proposed controller has two databases, routing, admission control, resource management modules, and three helper modules. The two databases are maintained in the background without interfering with operations performed by modules. Routing, admission control, and resource management modules perform main tasks, but they can cooperate with other helper modules.

Link State Database (LSDB) keeps track of physical network elements that are links and switches. Some of them are the identifier of the corresponding switch data-route identifier (DPID), port numbers, maximum transfer rate, power consumption, and link load. Note that link load is not a static value unlike others because it changes in time depending on the network load. Route State Database (RSDB) keeps track of PCRs and other information regarding them. This database is the virtualized version of the physical network. Controller mostly uses this database for network management purposes. RSDB contains both static and temporary data. Some of the static information is route identifier (RID), data-path identifiers of edge nodes (i.e., ISS and ESS), links that form the route. Some of the dynamic data are route capacity and route load.

Routing and Admission Control Module (RACM) routes flow between the associated edge nodes. In routing, the controller assigns the incoming flow to a route of the corresponding pair. In admission control, the controller checks available resources before deciding the routing. If there is an available resource, the controller allows routing of flow through assigning the flow to a route. Otherwise, it simply rejects the flow. This way, the controller prevents the overloading of resources. Load

Balancing Module (LBM) equalizes the cost of routes per pair. At first, it computes the route costs of the corresponding pair and calculates the equalization cost. Afterward, if shifts flow from overloaded routes to underloaded routes. Route Resizing Module (RRM) updates the capacity of routes depending on the load of neighbor routes and pairs. Whenever the load of the route exceeds a certain threshold, the resizing procedure is invoked. Energy Saving Module (ESM) aggregates flows into a smaller number of routes. Then, it puts unused network elements into sleep mode.

Route Management Module (RMM) establishes, updates, or removes routes PCRs. Information Collection Module (ICM) collects data such as link rates from underlying switches and performs necessary calculations on them to extract the information required for routing, admission control, load balancing, route resizing, and energy-aware resource management. Subsequently, all the information is stored in databases. Throughout this dissertation, we assume that there is a single controller residing on the control plane. However, there must be multiple controllers in the control plane to avoid single point of failure. Thus, the controller has Inter Controller Communication Module (ICCM) that is responsible for communicating with other controllers in the control plane.

4.4. Admission Control and Routing

In an SDN-based network, the switch which receives a request f_{req} for a new flow f sends a packet-in message to the controller. As soon as the controller receives this packet, it computes the shortest route between end nodes. Subsequently, it deploys forwarding rules off to the corresponding switches along the route. This may make the controller bottleneck as the number of flows and switches increases. Similar to the routing, the controller must check whether there is enough amount of resource (i.e., bandwidth) in the network to accept the request during the admission control process.

In the proposed model, the controller does not make per flow route computation. It simply assigns the new flow f to a route as soon as it gets f_{req}. This makes the routing decision independent from network size. Hence, routing scalability is achieved. Similar to the routing, the controller checks available resources of route between edge

pairs based on the information in RSDB. If there is enough amount of resource, routing of f is allowed. Otherwise, the controller rejects f_{req}.

The first step of both routing and admission control is the edge pair identification step. In this step, the controller already knows about ISS due to the packet-in message. It identifies the corresponding ESS by extracting the destination address from the packet-in message and querying RSDB. The next step is the assignment of the flow to one of the routes of the corresponding edge pair. We develop a flow assignment method (hash-based modulo-ρ assignment operation - HMA) where the assignment of a flow to a route is proportional to the capacity of routes. Finally, the controller deploys forwarding rules to edge nodes for routing. Any packet after this operation is forwarded on the assigned route.

In HMA, each route $P_k^{s,d}$ has an assignment range specified with $\delta_{min}^{s,d,k}$ and $\delta_{max}^{s,d,k}$. This range depends on the number of routes $|P^{s,d}|$ of edge pair, the capacity of each route $\varphi(P_k^{s,d})$, and the maximum range value ρ. The value of ρ can be any value greater than 0; that is, there is no upper limit. The network administrator can pick any positive value for ρ. Note that assignment ranges of routes of a pair do not change unless their capacity changes. Change in capacity happens in two cases. In the first case, the controller performs the resizing operation, which we will explain later in this section. In the second case, route re-establishment happens due to a link or switch failure. Route re-establishment can also happen in case of the desire of the network administrator. All of these imply that the controller does not frequently perform assignment range computation. The computation of the assignment range is defined below.

$$\delta_{min}^{s,d,k} = \begin{cases} 0 & k = 1 \\ \delta_{max}^{s,d,k-1} + 1 & 2 \leq k \leq |P^{s,d}| \end{cases} \quad \forall s \text{ and } d \in V, s \neq d. \tag{4.1}$$

$$\delta_{max}^{s,d,k} = \delta_{min}^{s,d,k} + \frac{\varphi(P_k^{s,d}) * \rho}{\sum_{k=1}^{|P^{s,d}|} \varphi\,(P_k^{s,d})}, \forall s \text{ and } d \in V, s \neq d \tag{4.2}$$

The minimum assignment value of a route is determined by its order as defined in Equation (4.1. For the first route, it is 0. For the others, it depends on the maximum

assignment value of the previous route. The maximum assignment value of a route depends on both ρ and proportion of the path capacity over pair capacity as defined in Equation (4.2. To be clearer, suppose that $\varphi(P_1^{1,1})$ and $\varphi(P_2^{1,1})$ in Figure 4.2 are 2 and 3Gbps, respectively. Total capacity ($\sum_{k=1}^{|P^{s,d}|} \varphi\,(P_k^{s,d})$) is 5Gbps. Assuming that ρ is 100, $\delta_{min}^{1,1,1} = 0$, and $\delta_{max}^{1,1,1} = 40$. Similarly, $\delta_{min}^{1,1,2} = 41$, and $\delta_{max}^{1,1,2} = 100$. As it is seen in the example, a route with the largest capacity has the highest assignment probability.

The complete HMA process works as follows. First, controller computes hash value H(f) of f_{req} using fields in packet-in message such as destination IP. Any hashing method (e.g., MD5 [75], SHA-1 [76]) can be exploited for this purpose. Subsequently, modulus ρ of H(f), denoted as ρ(f), is computed. The result value lies within the assignment range of a route (i.e., $\delta_{min}^{k,j} \leq \rho(f) \leq \delta_{max}^{k,j}$).

Table 4.2: Routing Algorithm.

Lines	**Steps**
	Input: f_{req}
	Output: Routing of f_{req}
1	$ISS = getSwitchDPID(f_{req})$
2	$ESS = getDestinationSwitchFromRSDB(f_{req})$
3	$\rho(f) = calculateHMA(f_{req})$
4	$P^{s,d} = getRoutesOfPair(ISS, ESS)$
5	**FOR** k **IN** $P^{s,d}$:
6	**IF** $\delta_{min}^{s,d,k} \leq \rho(f) \leq \delta_{max}^{s,d,k}$
7	$assignToRoute(f_{req}, P_k^{s,d})$
8	**END IF**
9	**END FOR**

Both routing and admission control algorithms are presented in Table 4.2 and Table 4.3, respectively. In Table 4.2, controller identifies edge pairs as soon as it receives flow request f_{req}. Afterwards, it performs HMA operation to determine forwarding route. Finally, controller deploys forwarding rules to edge nodes. Admission control algorithm resembles routing algorithm. Apart from Table 4.2,

controller checks if there is available resource in edge pair before HMA operation in Table 4.3. After flow assignment for both routing and admission control, forwarding of packets starts in data plane. First, ISS marks incoming packets of f with an associated RID. Subsequently, it forwards marked packets to the related core switch. Then, marked packets are forwarded through the core network and they arrive to the ESS. Finally, ESS removes marks on packets and delivers them to the destination node or another network.

Table 4.3: Admission Control Algorithm.

Lines	**Steps**
	Input: f_{req}
	Output: Acceptance/rejection of f_{req}
1	$\mathrm{ISS} = \mathrm{getSwitchDPID}(f_{req})$
2	$\mathrm{ESS} = \mathrm{getDestinationSwitchFromRSDB}(f_{req})$
3	**IF** $!\,\mathrm{isResourceAvailable}(f_{req}, \mathrm{ISS}, \mathrm{ESS})$:
4	$\quad \mathrm{reject}(f_{req})$
5	**END IF**
6	$\rho(f) = \mathrm{calculateHMA}(f_{req})$
7	$P^{s,d} = \mathrm{getRoutesOfPair}(\mathrm{ISS}, \mathrm{ESS})$
8	**FOR** k **IN** $P^{s,d}$:
9	$\quad$ **IF** $\delta_{min}^{s,d,k} \leq \rho(f) \leq \delta_{max}^{s,d,k}$
10	$\quad\quad \mathrm{assignToRoute}(f_{req}, P_k^{s,d})$
11	$\quad$ **END IF**
12	**END FOR**

4.5. Load Balancing

Load balancing is a fundamental resource utilization operation. Load balancing aims to distribute traffic load over the network fairly. This in turn reduces both capital expenditures (CAPEX) and operational expenditures (OPEX). In addition to this, HMA necessitates load balancing because flows of a pair may be assigned to only

some of its routes. This results in the under-utilization of the remaining routes. Therefore, overall network utilization reduces.

The proposed controller performs load balancing adaptive to the traffic for two reasons. First, the controller has limited computational resources. Thus, computationally intensive operations such as optimal (network-wide) load balancing may degrade the performance of the controller in the case of scalability. Instead of optimizing global network load, the controller performs pair-based load balancing (local). Local load balancing converges to network-wide load balancing in time since route costs reflect physical link costs. Secondly, optimum load balancing does not last a long time because traffic is highly dynamic.

In our model, the controller balances load of a pair among its routes by equalizing their costs (i.e., load, utilization, congestion level). In the first step of load balancing, controller computes route costs and an equilibrium cost. Equilibrium cost is the cost that all route costs must be closer after load balancing operation. Secondly, the controller determines route states (i.e., underloaded, balanced, overloaded) with respect to the equilibrium cost. Finally, the controller shifts some of the flow in overloaded routes to underloaded routes for cost equalization. Shifting is as simple as updating the forwarding rule of the flow in edge switches.

Notice that a route consists of one or more links. In that sense, link costs must reflect route costs. Denoting the capacity and load of a physical link as $\varphi(\mathrm{E_i})$ and $\vartheta(\mathrm{E_i})$, respectively, the controller calculates link cost based on the equation defined below.

$$\lambda(\mathrm{E_i^{s,d,k}}) = \frac{\mathrm{Q_i}}{1 - \frac{\vartheta(\mathrm{E_i}) - 1}{\varphi(\mathrm{E_i})}}, \forall \mathrm{i} \in \mathrm{E^{s,d,k}}, \mathrm{s} \neq \mathrm{d} \tag{4.3}$$

First, link utilization is computed by dividing $\vartheta(\mathrm{E_i}) - 1$ to $\varphi(\mathrm{E_i})$. Subtracting one from $\vartheta(\mathrm{E_i})$ prevents value of denominator to be 0 in the Equation (4.3 when link is fully loaded. After link utilization calculation, the controller subtracts the link utilization value from 1, and divides Q_i with it. Linear increment in $\vartheta(\mathrm{E_i})$ results in exponential increment in $\lambda(\mathrm{E_i^{s,d,k}})$. In other words, link cost becomes sensitive to load changes. This way, the model avoids overloaded links. We introduce the coefficient Q_i to the equation so that network administrators can adjust link costs just in case.

Also, note that the controller does not compute link cost per load balancing operation. Instead, the controller maintains link and route costs in the LSDB, and RSDB, respectively, distinct from all operations.

We define route cost as the summation of costs of links which form the corresponding route. Denoting the cost of a route $P_k^{s,d}$ as $\lambda(P_k^{s,d})$, route cost computation is defined as follows:

$$\lambda(P_k^{s,d}) = \sum_{i=1}^{|E^{s,d,k}|} \lambda\left(E_i^{s,d,k}\right), \forall k \in P^{s,d}, s \neq d \tag{4.4}$$

Load Balancing Algorithm is presented in Table 4.4. In the first step, it calculates equilibrium cost of the pair, denoted as $\mu(V_s, V_d)$, considering routes. Cost of all the routes converge to $\mu(V_s, V_d)$ after load balancing process. In the second step, state of routes is identified based on $\mu(V_s, V_d)$. A route whose cost is greater than $\mu(V_s, V_d)$ is classified as overloaded. Similarly, a route whose value is less than $\mu(V_s, V_d)$ is classified as underloaded. If a route has the same cost as $\mu(V_s, V_d)$, it is already balanced. Thus, controller does not take this route into account. In the third step, the algorithm obtains information such as ISS, ESS and RID regarding each flow in unbalanced routes. Finally, some these flows are shifted in overloaded routes to underloaded routes. Controller selects which flow to be shifted in a greedy manner (e.g., heavily to lightly loaded flow).

Table 4.4: Load Balancing Algorithm.

Lines	**Steps**
	Input: f_{req}
	Output: Equalization of pair load
1	$\text{pairCost} = 0$
2	**FOR** Route P_k in $P^{s,d}$:
3	**IF** $\text{isActive}(P_k)$:

The computation of pair equilibrium cost takes $\Theta(n)$ in which n is the number of routes per pair. Sorting flows by size takes $O(f \log(f))$ in which f is number of flows. Shifting flows to underutilized routes takes $O(n \log(f))$. Thus, time complexity of the

Active Load Balancing algorithm is $O\left(n\left(f\ \log(f) + (n\ f)\right)\right)$ in the worst case where all routes are active. The controller performs load balancing for all pairs denoted by B. Therefore, time complexity for overall network load balancing becomes $O(B\ n\ (f \log(f) + n\ f))$.

Table 4.5: Load Balancing Algorithm (Contd.).

4	pairCost = pairCost + $\lambda(P_k)$
5	**END IF**
6	**END FOR**
7	$\mu((V_s, V_d))$ = pairCost / $\lvert P^{s,d} \rvert$
8	**FOR** Route P_k **IN** $P^{s,d}$:
9	**IF** ! isActive(P_k):
10	continue
11	**END IF**
12	**IF** $\lambda(P_k) < \mu((V_s, V_d))$:
13	addRouteToList(lightlyUtilizedRouteList, P_k)
14	**ELSE IF** $\lambda(P_k) > \mu((V_s, V_d))$:
15	addRouteToList(highlyUtilizedRouteList, P_k)
16	**END IF**
17	**END FOR**
18	**FOR** Route P_o **IN** highlyUtilizedRouteList:
19	flowPool = getFlows(P_o)
20	sortFlowsByAscendingLoad(flowPool)
21	**FOR** Route P_u **IN** lightlyUtilizedRouteList:
22	**FOR** Flow f_a **IN** flowPool:
23	**IF** shifting flow f_a to P_u makes $\lambda(P_u) < \mu((V_s, V_d))$:
24	shiftFlow(f_a, P_o, P_u)
25	**END IF**
26	**IF** $\lambda(P_u) > \mu((V_s, V_d))$:
27	addRouteToList(routeRemoveList, P_u)
28	continue

Table 4.6: Load Balancing Algorithm (Contd.).

29	**END IF**
30	**END FOR**
31	**END FOR**
32	**FOR** Route P_r **IN** routeRemoveList:
33	removeRoute(routeRemoveList, P_r)
34	**END FOR**
35	**IF** $\lambda(P_o) < \mu((V_s, V_d))$:
36	continue
37	**END IF**
38	**END FOR**

There are two types of load balancing initiation in the proposed model. That are periodic and threshold based. In the periodic load balancing, the controller periodically performs load balancing such as per minute. How to determine the period is crucial.

4.6. Route Resizing

In the proposed model, PCRs are virtual, so their capacities are re-sizable. Dynamically adjusting route capacities prevents the network from congestion and increases resource utilization. Considering the pairs in Figure 4.1, the pair (V_2, V_1) may suffer from congestion while (V_1, V_1) is lightly loaded. However, (V_2, V_1) would not suffer if (V_1, V_1) could give some of its capacity to its neighbor. To overcome this problem, we propose a route resizing method to adjust route capacities whenever a route exceeds a certain threshold (e.g., 80% of its maximum capacity). Note that, this process is performed on RSDB, not in the data plane. For this reason, the controller does not explicitly alter the rule of routes on switches.

As aforementioned, neighbor routes may share physical links. A physical link is split into one or more virtual links where each of which serves a different route from point of view of the controller. Each virtual link $E_i^{s,d,k}$ is a part of a route $P_k^{s,d}$, where the triple (s, d, and k) of each virtual link is different. We denote a route $P_k^{s,d}$ on a link E_i as $P_{s,d,k}^i$. RRM computes new virtual link capacity $\varphi(E_i^{s,d,k})$ for all routes $P_{s,d,k}^i$

proportional to their loads, denoted as $\vartheta(P_{s,d,k}^{i})$. The corresponding equation is provided below.

$$\varphi\left(E_i^{s,d,k}\right) = \varphi(E_i) \times \frac{\vartheta\left(P_k^i\right)}{\vartheta(E_i)}, \forall s \; and \; d \in V, k \in P^{s,d}, i \in E, s \neq d \tag{4.5}$$

As soon as the controller performs this computation per virtual link, the new route capacity becomes the capacity of one of its virtual links which have the minimum capacity. To be clearer, consider a scenario based on the network illustrated in Figure 4.2. Let $\varphi(P_2^{1,1})$ and $\varphi(P_1^{2,1})$ be 6 and 4Mbps, respectively. Let also $\vartheta(P_2^{1,1})$ and $\vartheta(P_1^{2,1})$ be 5.7 and 1Mbps, respectively. In this case, $P_2^{1,1}$ exceeds the threshold, assuming that it is 80%, and the controller initiates the resizing process. Therefore, new $\varphi(P_2^{1,1})$ and $\varphi(P_1^{2,1})$ become 8.5 and 1.5Mbps, respectively.

Table 4.7: Route Capacity Resizing Algorithm.

Lines	**Steps**
	Input: Links
	Output: New capacities of all virtual links
1	**FOR** Link E_i in E:
2	**IF** isActive(E_i):
3	$\varphi(E_i)$ = getLinkCapacityFromLSDB(E_i)
4	$\vartheta(E_i)$ = getCurrentLinkLoadFromLSDB(E_i)
5	**FOR** Route P_k^i < **IN** getRoutesFromRSDB(E_i):
6	**IF** isActive(P_k^i):
7	$\vartheta(P_k^i)$ = getCurrentRouteLoadFromRSDB(P_k^i)
8	$\varphi(E_i^{s,d,k}) = \varphi(E_i)\, \vartheta(P_k^i) \,/\, \vartheta(E_i)$
9	**END IF**
10	**END FOR**
11	**END IF**
12	**END FOR**
13	**FOR** Route P_k **IN** $P^{s,d}$:

Table 4.8: Route Capacity Resizing Algorithm (Contd.).

14	**IF** ! isActive(P_k):
15	continue
16	**END IF**
17	linksOfRouteList = getLinksFromRSDB($P_k^{s,d}$)
18	temporaryMinimumCapacity = Maximum Link Capacity
19	**FOR** Link $E_i^{s,d,k}$ **IN** linksOfRouteList:
20	**IF** isActive($E_i^{s,d,k}$) **AND** temporaryMinimumCapacity > $\varphi(E_i^{s,d,k})$:
21	temporaryMinimumCapacity = $\varphi(E_i^{s,d,k})$
22	**END IF**
23	$\varpi(P_k^{s,d})$ = temporaryMinimumCapacity
24	**END FOR**
25	**END FOR**

Although the computation of the new $\varphi(E_i^{s,d,k})$ is simple, the time complexity of the resizing algorithm is $O(|E||P|^2)$. However, RRM does not perform this operation frequently unless the overall network is heavily loaded. In a heavily loaded network (e.g., 80% of its total capacity) the controller performs consecutive resizing operations because the threshold is exceeded even after the resizing operation. This infers that route resizing must be avoided when the network load is high. To handle this problem, the proposed controller stops performing on-demand resizing if the number of on demand resizing operation exceeds a predefined threshold in a predefined time interval such as 3 times in 3 minutes. It simply waits for some back-off time and then starts on-demand resizing. As it is noticed, RRM works on a virtual network during this operation because routes and their capacities are virtual. It does not communicate with underlying network elements.

4.7. Signaling

In SDN-based networks, the controller must communicate with underlying switches to manage the whole network in SDN. The number of control messages increases as the size of the network increases. Thus, heavy control traffic may occur

between controller and switches, especially in the case of a high number of flows and switches in a network.

In terms of routing and admission control signaling, usually, an SDN controller sends forwarding rules to all the switches along the route per flow. Therefore, the number of messages to be sent becomes the number of switches along the route per flow. Compared to this, the proposed controller sends only two messages independent of the number of switches along the route, that are ingress and egress switches. Thus, signaling scalability is achieved regarding routing and admission control.

As the proposed controller requires an up-to-date network state to manage the whole network, it updates link states periodically. Edge switches are busy with sending flow requests to the controller, inserting new flow rules to their tables, and updating existing rules during the load balancing process. For these reasons, the proposed controller does not communicate with edge switches to get link states. Instead, it collects link states from the corresponding core switches.

Considering the load balancing process, suppose that there are no PCRs, the controller treats the flow to be shifted as a new flow and sends forwarding rules to new switches between source and destination. Besides, old flow rules are deleted to maintain switch tables. Therefore, signaling scalability issues may arise as the number of flows to be shifted, and network size increases. In the proposed model, the controller sends a flow update message to an ingress switch for shifting a flow from one route to another during the load balancing process. Edge switches only update their output ports to shift a flow from one route to another.

5. ENERGY AWARE RESOURCE MANAGEMENT

There occurs a trade-off between optimal energy saving and computation overhead. In today's highly dynamic Internet traffic, the controller may become a bottleneck while computing optimal energy saving. Additionally, the optimal energy state may not last long due to the traffic fluctuations. Thus, we prioritize scalability over optimal energy saving. To do so, our controller makes edge pair-based energy-saving computation. Upon completion of pair-based optimization, the controller aggregates pair traffic into designated routes. Subsequently, it deactivates idle links and switches in the whole network. This local energy saving converges to a global energy saving in time.

To be clearer about how energy can be saved with traffic aggregation over a smaller number of routes, consider the network illustrated in Figure 4.1. The routes $P_2^{1,1}$ and $P_1^{2,1}$ are neighbors because they share the links E_6 and E_9. Assuming that these routes have large enough capacity to carry their pair loads, the controller aggregates whole network traffic to $P_2^{1,1}$ and $P_1^{2,1}$. Then, it deactivates the remaining 4 core switches and 6 links to save energy.

The first step of energy-aware route management is determination of the most energy efficient subset of routes of a pair $P^{s,d}$ while total capacity of $P^{s,d}$ ($\varphi(P^{s,d})$) is greater than or equal to its load ($\vartheta(P^{s,d})$). Time complexity of the brute force solution (generation of all combinations ($\theta(2^n)$) and searching for the best combination ($\theta(n)$)) is exponential. In this regard, we propose a polynomial time algorithm called Energy-aware Route Management Algorithm (ERMA) defined in Table 5.2: Energy Saving Algorithm.

Let $\varphi(P^{s,d})$ denote the pair capacity, $\vartheta(P^{s,d})$ denote the pair load and $V[i,c]$ ($1 \leq i \leq n$ and $\vartheta(P^{s,d}) \leq c \leq \varphi(P^{s,d})$) denote the minimum energy consumption for a subset of routes for number of i route and c capacity. Assuming that all the routes are initially active, ERMA decides if the i-th route should be active or passive, at each step. In the former case, $V[i,c]$ is equal to $V[i-1,c]$. This means energy consumption and capacity stays the same. In the latter case, $V[i,c]$ becomes $V[i-1,c-c_i]-e_i$. More clearly, current capacity c reduces by c_i and energy consumption reduces by e_i.

We can find the optimal solution by filling the table $V[0..n, 0..\varphi(P^{s,d})]$. In this sense, entry $V[n, \vartheta(P^{s,d})]$ becomes the optimal solution. According to our problem, $V[i, c] = \infty$ for $0 \leq c \leq \vartheta(P^{s,d})$ and $V[0, c]$ is equal to maximum energy consumption for $0 \leq c \leq \vartheta(P^{s,d})$.

$$PDV\left(P_k^{s,d}\right) = \begin{cases} 2^{\left(PP - RUR\left(P_k^{s,d}\right)\right) * \frac{10}{PP}}, & \text{if } RUR\left(P_k^{s,d}\right) \leq PP \\ 2^{\left(RUR\left(P_k^{s,d}\right) - PP\right) * \frac{10}{100 - PP}}, & \text{if } RUR\left(P_k^{s,d}\right) > PP \end{cases} \quad (5.1)$$
$$, \forall s \; and \; d \in V, s \neq d, k \in P^{s,d}$$

The controller computes the energy consumption value of a route $P_k^{s,d}$ by multiplication Passive Neighbor Ratio (PNR), and Pivot Distance Value (PDV). PNR and PDV of a route $P_k^{s,d}$ is indicated as PNR($P_k^{s,d}$) and PDV($P_k^{s,d}$), respectively. PNR($P_k^{s,d}$) is the ratio of passive neighbor routes over all of its routes. PDV($P_k^{s,d}$), however, depends on both route utilization ratio $\left(RUR\left(P_k^{s,d}\right) = \vartheta\left(P_k^{s,d}\right) * \frac{100}{\varphi\left(P_k^{s,d}\right)}\right)$ and pivot point (PP) as defined in Equation (5.1. PP is a predefined utilization ratio (e.g., 70%). It allows the controller to aggregate the desired amount of load to the route as much as possible. For instance, defining PP as 70% forces the controller to make the route utilization around 70% but at the same time prevent it from over-utilization. According to the Equation (5.1, PDV($P_k^{s,d}$) changes exponentially from the distance between route utilization and PP. The reason is to force the controller to reach the desired utilization as fast as possible.

Time complexity of ERMA is $\mathcal{O}(n * |\varphi(P^{s,d}) - \vartheta(P^{s,d})|)$. Notice that we design ERMA for a single pair. Therefore, total time complexity for the whole network becomes $\mathcal{O}(n * |\varphi(P^{s,d}) - \vartheta(P^{s,d})| * B)$ where B is number of pairs. As soon as the controller determines future active and passive routes, it may need to shift flows from the routes that will be passive to the routes that will remain active or just have been activated. In such cases, the controller determines current and new states of the routes first. To be clear, there are four states which are defined in Table 5.1. If there are routes

Table 5.1: Possible Energy State Changes of Routes.

States	**Current State**	**Next State**
State 0	Active	Active
State 1	Active	Passive
State 2	Passive	Active
State 3	Passive	Passive

in State 1, the controller shifts the flows in these routes to the routes that are either in State 0 or State 2.

After the shifting process, the controller tries to deactivate all the network devices along the routes that do not share switches and links along their way with other routes. Since the whole route is passive, energy-saving increases dramatically. If this is not the case, the controller tries to deactivate the switches and links along its way that does not carry traffic. Before deactivating a switch, the controller makes sure that all the links coming and leaving the switch does not carry traffic.

ERMA finds the optimum route combination per pair assuming that all the routes are active. However, some of the routes may be in passive mode in reality and their capacity may be taken by neighbor routes. To handle this problem, ESM computes future capacities of passive routes because a passive route can be selected as a result of ERMA. To do so, it calculates the unused capacity of each link by subtracting load from its capacity and share this unused capacity equally among the passive routes that the link serves. The capacity of the passive route becomes the capacity of the link which has the minimum capacity compared to other links that serve the same passive route. Let us show this procedure in an example for clarification purposes. Suppose that ESM executes ERMA for the pair ISS_1-ESS_1 as illustrated in Figure 4.1. Also, suppose that capacities of all links are 1Gbps and routes except $P_2^{1,1}$ and $P_1^{2,1}$ are active. The controller must find the future capacity of Route2 in case ERMA selects it. There are 1, 2 and 2 passive routes on links E_2, E_6 and E_9, respectively. Load of E_2, E_6 and E_9 are all 0 because $P_2^{1,1}$ and $P_1^{2,1}$ do not carry traffic. Unused capacities of E_2, E_6 and E_9 are 1Gbps for each. In this case, capacity share of $P_2^{1,1}$ for the links E_2, E_6 and E_9 are 1, 0.5 and 0.5Gbps, respectively. The minimum of these capacities is 0.5Gbps. Therefore, the future capacity of $P_2^{1,1}$ becomes 0.5Gbps.

Table 5.2: Energy Saving Algorithm.

Lines	Steps
	Input: Pair load $\vartheta(P^{s,d})$, pair capacity $\varphi(P^{s,d})$, pair energy $\varepsilon(P^{s,d})$
	Output: new states for paths of the pair
1	**FOR** $i = 1$ **IN** n:
2	$V[i, \varphi(P^{s,d})] \Leftarrow \varepsilon(P^{s,d})$
3	**END FOR**
4	**FOR** $i = 1$ **IN** n:
5	**FOR** $c = \varphi(P^{s,d})$ **TO** $\vartheta(P^{s,d})$:
6	**IF** $c - c_i \geq \vartheta(P^{s,d})$ **AND** $(V[i-1,\ c - c_i] - e_i) \leq (V[i-1, c])$:
7	$V[i, c] \Leftarrow V[i-1, c - c_i] - e_i$
8	**ELSE**:
9	$V[i, c] \Leftarrow V[i-1, c]$
10	**END IF**
11	**END FOR**
12	**END FOR**

6. IMPLEMENTATION DETAILS

6.1. Databases

The proposed controller maintains Link-state Database to keep track of links and the whole network topology. Basically, they are identifier of L_i, switch data-path identifiers (DPID) and ports that L_i connects, $\varphi(L_i)$, rate of L_i and $\lambda(L_i^k)$. The latter two information changes in time-based on the traffic passes through the link.

In addition to LSDB, the proposed controller maintains Route-state Database to keep track of PCRs. In the proposed model, we assume that routes are pre-established. For this reason, we provide route information to the controller via a network definition file NDF instead of performing a multi-route routing algorithm. During the lifetime of PCRs, the controller maintains this database whenever any route related information changes such as route costs. Since we determine route cost based on link costs, the controller updates RSDB whenever link costs in LSDB are updated. Some static information of a route maintained in RSDB is its identifier (RID), DPIDs of ISS and ESS, links that form the route. RSDB also has dynamic data that are route capacity and $\lambda(P_k^j)$.

6.2. Route Management Module

We implemented the Route Management Module so that the proposed controller can create, update, and remove them. In this part of the section, we will explain implementation details of PCRs and packet forwarding over routes in the data plane.

6.2.1. Route Establishment

At the beginning of the route establishment process, the controller parses NDF to learn about the details of routes. In our implementation, we used Virtual LAN (VLAN) to establish routes using OpenFlow v1.0 [20]. Each VLAN has a globally distinct RID. The RID is just an integer value. After parsing the definition file, the controller sends OFP_FLOW_MOD_ADD messages to the corresponding core

switches using the Static Flow Pusher Module of Floodlight. Note that there are 4094 unique routes because VLANs in 802.1Q use 12 bits in which 0 and 4095 are reserved. MPLS can be used for routes instead of VLAN but this requires OpenFlow v1.3 [77] or higher versions.

Controller populates only tables of core switches with route rules during the route establishment process. In other words, there is no specific rule for a single flow in a CSS. For instance, consider the route Route1 in the Figure 4.1. It consists of ISS, CSS11, CSS21, and ESS. According to this example, the controller sends route establishment messages only to CSS11 and CSS21. Edge switches (e.g., ISS1 and ESS) are empty. Thus, there is no full route.

Let us explain how we create a complete route for a flow f. Suppose that f comes to ISS_i for the first time and it should depart the network from ESS_j. In this case, the establishment of full route for f occurs as follows:

i) ISS_i receives the first packet of flow f.
ii) ISS_i looks for a match with the packet regarding to existing forwarding rules. Since flow is new, incoming packets do not match with any of the existing flow rules.
iii) ISS_i prepares an OFP_PACKET_IN message and sends it to the controller
iv) The controller extracts destination address of packets from the OFP_PACKET_IN message and determines the corresponding egress switch which is ESS_j.
v) The controller queries the RSDB to obtain routes of the pair ISS_i - ESS_j.
vi) The controller determines the route for f in HMA step.
vii) The controller prepares OFP_FLOW_MOD_ADD message for ISS_i. In the match part of the OF message, there are source and destination IP addresses. In the action part of the same message, there are two actions. The first action marks the packet with associated RID specified in OFPAT_SET_VLAN_VID action. The second action sends the packet to the core switch from the physical port defined in the OFP_ACTION_OUTPUT action. The same applies for egress, but this time, the controller exchanges source and destination addresses.
viii) The controller prepares another OFP_FLOW_MOD_ADD message for ESS_j. In the match part of the OF message, source and destination IP addresses are

swapped compared to the message described in the previous step. In the action part of this message, there are two actions. The first action removes the RID with OFPAT_STRIP_VLAN action. The second action sends the packet out of network from the physical port defined in the OFP_ACTION_OUTPUT action.

Note that distance between an edge pair can be one hop. Even in this case, controller generate and install routing rules described as above.

6.2.2. Packet Forwarding

Packet forwarding starts in an ISS and ends in an ESS after the controller installs flow rules to the corresponding edge switches. Forwarding of a packet of a flow f coming from ISS_i and destined to ESS_j is performed as follows:

i) The packet arrives at ISS_i and ISS_i looks for a matching rule regarding f.
ii) ISS_i sets VID part of the 802.1Q header (within Ethernet frame) of the packet with OFP_SET_VLAN_VID action of the associated rule to assign the packet to a route.
iii) ISS_i forwards the packet from the out-port specified in the actions of the same rule.
iv) The first core switch, which is the neighbor of ISS_i, receives the packet. It finds the matching route rule by checking the RID of the packet. Subsequently, it forwards the packet to the next core switch along the route.
v) Each core switch along the route forwards the packet the same way as the first core switch and the packet eventually reaches ESS_j.
vi) ESS_j takes the packet and removes the VID of the packet by OFP_STRIP_VLAN action. Finally, it forwards the packet out of the network over the specified out-port.

The response message for the packet is routed back in the same way. However, this time, egress acts as ingress and vice versa.

6.3. Routing and Admission Control Module

If a packet does not match with any rules in an OF switch, the packet is called unknown. As soon as an OF switch receives such a packet, it sends a message, namely OFP_PACKET_IN, to the controller as a default action in OF v1.0. As soon as the controller receives the OFP_PACKET_IN message, it takes an action such as creating a new route or rejecting the request.

In our implementation, only ingress switches send such requests to the controller due to the structure of SPNs. As soon as the controller receives a request from an ISS, RACM steps in. Briefly, RACM handles admission control in two steps. First, it determines the corresponding ingress-egress pair based on source and destination IP addresses. Secondly, RACM queries RSDB to check if there exists enough amount of bandwidth. If so, the controller assigns the flow to a route and sends new rules to associated edge switches. Otherwise, the request is rejected.

6.4. Information Collection Module

The controller only needs link loads for resource management operations. In OpenFlow, port statistics provide cumulatively transmitted and received byte counts. The Information Collection Module periodically sends OF port statistics message to related core switches per link. Upon receiving OF reply messages from the CSSs, the controller computes rates of links as follows:

i) Controller subtracts previously obtained byte counts from current ones.
ii) The result of subtraction is divided by the time interval between consecutive measurements.

Therefore, ICM obtains approximate link rates. Consequently, the controller calculates link costs as described in the Equation (4.3. Finally, ICM updates LSDB in the background.

6.5. Load Balancing Module

In the proposed model, load balancing has a key role to increase resource utilization. The controller can initiate the load balancing process in two ways. In one way, LBM periodically balances each pair (e.g., every 5 minutes). In another way, the controller checks the cost difference between all routes of a pair. If the difference between any two routes exceeds a certain threshold, LBM initiates the load balancing process. The threshold value can be constant or adaptive. LBM follows the steps below during the load balancing process of a pair.

i) LBM retrieves route costs from RSDB.
ii) LBM calculates average pair cost.
iii) LBM calculates difference between average pair cost and route costs per route.
iv) If the result is positive for a route, LBM marks the route as heavily loaded. If not, it marks the route as lightly loaded.
v) LBM obtain flow statistics from the corresponding ISS for overloaded routes.
vi) LBM selects subset of flows in a greedy manner (i.e., lightly loaded to heavily loaded). It stops when cost of lightly loaded route is equal or within the pre-defined distance by the network administrator from the average pair load.
vii) LBM sends route update message for each flow in the subset.

6.6. Route Resizing Module

Route Resizing Module is responsible for updating virtual route capacities. RRM periodically accesses RSDB to check route utilization. If a route exceeds a certain threshold, RRM invokes the resizing process.

RRM does this operation in three steps. First, it computes virtual route capacity portions for each link. Results are stored in a matrix called Route-Link Matrix. In this matrix, rows are routes and columns are links. As soon as RRM fills the whole matrix, it sets the virtual capacity of each route to the minimum value of the row which is the minimum virtual capacity portion.

Table 6.1: Route-link Matrix of Virtual Network.

	L1	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**	**L9**	**L10**
$\mathbf{P_1^1}$	10	∞	∞	∞	10	∞	∞	10	∞	∞
$\mathbf{P_2^1}$	∞	10	∞	∞	∞	6.25	∞	∞	6.25	∞
$\mathbf{P_1^2}$	∞	∞	10	∞	∞	3.75	∞	∞	3.75	∞
$\mathbf{P_2^2}$	∞	∞	∞	10	∞	∞	10	∞	∞	10

To be clearer, consider the Table 6.1. There is a Route-Link Matrix for the topology illustrated in Figure 4.2. In this topology, there are 10 links and 4 routes. Whenever RRM invokes the resizing process, it iteratively calculates each cell in the matrix. In this example, the capacity of P_1^1 and P_2^2 does not change because there is no link shared with other routes. Thus, the controller does not perform any calculation for them. However, P_2^1 and P_1^2 shares L6 and L9. Suppose that capacity of all the links are 10Mbps and loads of P_2^1 and P_1^2 are 5 and 3, respectively. New capacities of P_2^1 and P_1^2 for L6 and L9 becomes 6.25 and 3.75, respectively.

Route resizing enables to increase admissions of more flow requests and reduces congestion. However, it becomes heavier as the number of links and routes in the network increase. For this reason, our controller resizes routes on demand (e.g., exceeding 80% of route size). Although this prevents the controller to perform resizing more than necessary, the controller may perform resizing consecutively. This happens when the threshold is exceeded even after the previous route resizing operation. To avoid this case, we introduce a periodic resizing parameter. If the number of on-demand resizing exceeds a certain threshold in a specific time interval (e.g., 3 times in a minute), the controller switches to periodic resizing mode. In this mode, the controller performs resizing periodically (e.g., 3 minutes). In periodic resizing mode, the controller continues to access RSDB to check whether it is still necessary to perform periodic resizing. If there is no need to perform periodic resizing, the controller switches to on-demand resizing mode. Additionally, notice that route resizing is a virtual operation. Thus, the controller and databases are the parts of this operation. Physical links are not affected by route resizing so signaling protocol is not used.

6.7. Energy Saving Module

Energy Saving Module is responsible for activating/deactivating links and switches if possible. To do so, the controller obtains required data from databases and computes route states. Accordingly, it activates or deactivates links or switches. Currently, OpenFlow is not capable of commanding a switch to sleep or wake up any of its ports. We propose to modify the OFPT_PORT_MOD message and OFP_PORT_CONFIG flag for this purpose. OFP_PORT_CONFIG has an OFPPC_PORT_ACTIVE flag that commands the switch to deactivate its corresponding physical port. By default, the flag OFPPC_PORT_ACTIVE on the switch port is set to 1. This implies that the corresponding physical port is active. The controller sends OFPT_PORT_MOD to toggle (active to passive or vice versa) the state of a port.

6.8. Inter Controller Communication Module

This module is responsible for communicating with other controllers in the control plane. This module is required for preventing the network from a single point of failure issue. This is the issue when there is a single controller in the whole control plane. When the single controller stops working, all the routing, admission control, and resource management operations stop. In this work, we do not address this issue as there are proposed works mentioned in [51] regarding this. A simple solution for our model is that there can be two or more controllers within the control plane. One of them is the master. The other controllers, a.k.a. backup controllers, periodically check if the master controller is alive. In case of failure of the master controller, one of the backup controllers takes over. As the proposed model maintains network state in databases, the backup controller can directly connect to LSDB and RSDB to manage the whole network. Thus, there is a minimal overhead of taking control in case of controller failure.

7. EXPERIMENTAL RESULTS

This section presents the experimental results of our model. We have implemented modules of our model upon Floodlight v0.91 [16]. For creating test networks, we have used both Mininet v2.1 [78] and Open vSwitch v2.5 [79]. Controller and switches communicate via OpenFlow v1.0 [20]. We used TG [80] as a traffic generator. Throughout this section, several scenarios under three topologies are tested to evaluate the proposed model. These topologies are NSFNET T3, Net-M, and Net-L as illustrated in Figure 7.1, Figure 7.2, and Figure 7.3. NSFNET T3 is a real network and is chosen as it provides connectivity to several regional networks [81], [82]. Apart from the NSFNET T3, two custom topologies, namely Net-M and Net-L are also used to test the proposed model in terms of topology size, the number of disjoint routes, connectivity among edge and core switches.

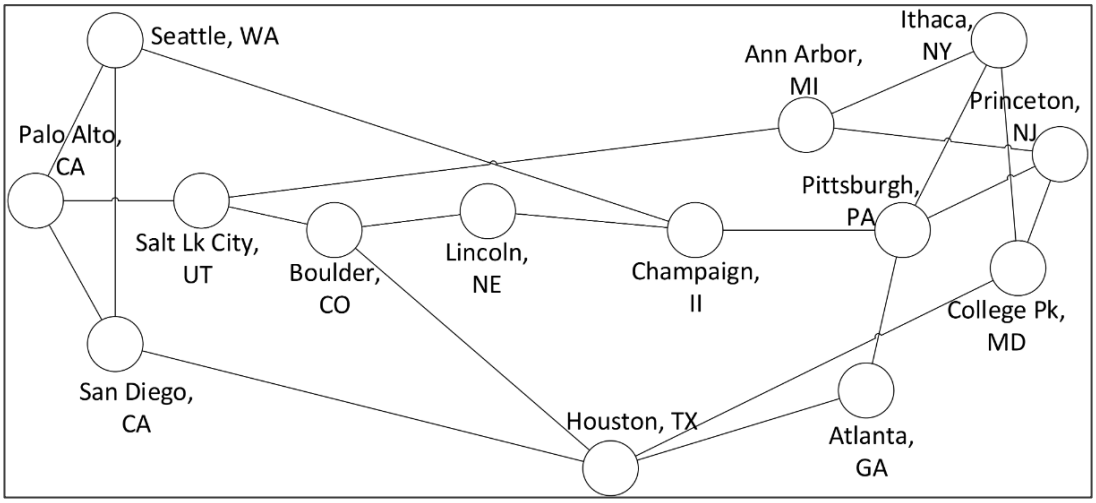

Figure 7.1: Illustration of NSFNET topology.

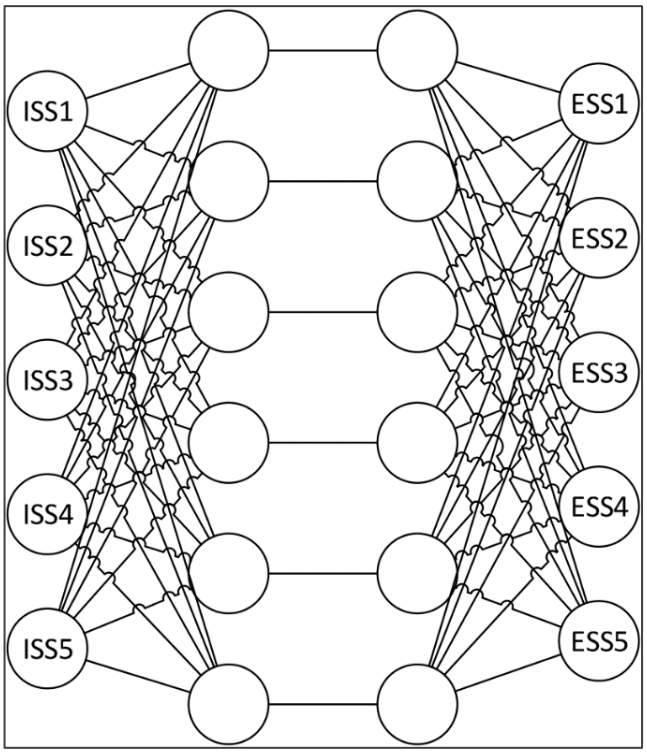

Figure 7.2: Illustration of Net-M.

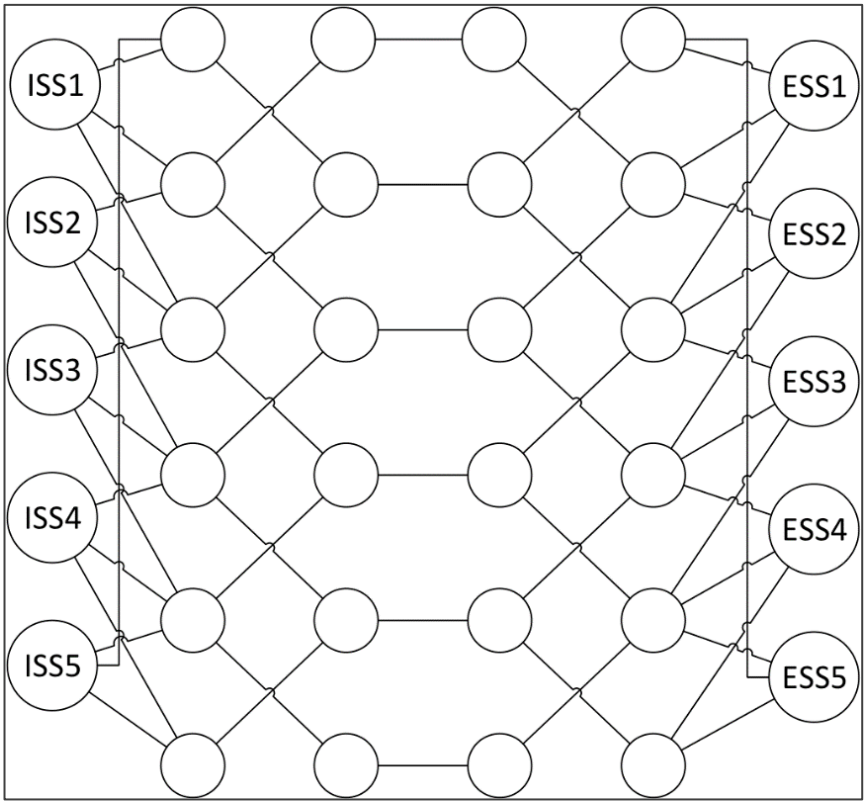

Figure 7.3: Illustration of Net-L.

In order to create a realistic test environment, several real-time traffic traces are taken from the Center for Applied Internet Data Analysis (CAIDA) [83]–[86] as depicted in Figure 7.4. We have generated various traffic loads (i.e., light, moderate, heavy) by scaling these traffic traces and assigned one of four traffic patterns arbitrarily to each pair in each test run.

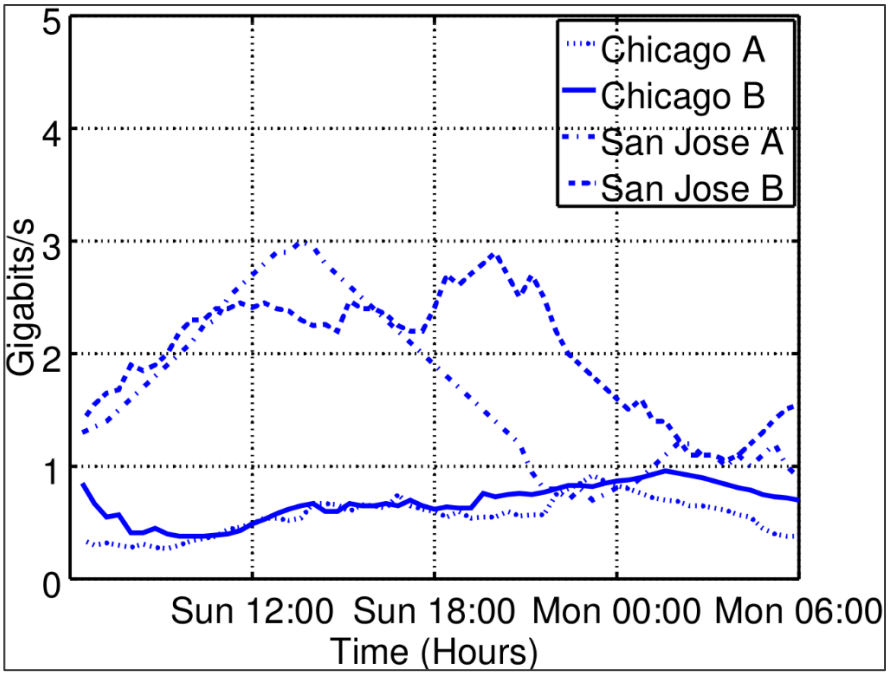

Figure 7.4: Traffic pattern of March 29-30 2015 UTC taken from CAIDA.

The test scenarios are performed to analyze; (1) effect of route number on load balancing performance, (2) periodic load balancing, (3) threshold-based load balancing, (4) load balancing performance under various traffic loads, (5) load balancing for different topologies, (6) comparison of the proposed model with the work proposed in [87], (7) admission control time, (8) effect of route number and traffic load on energy-saving performance, (9) effect of node connectivity on energy-saving performance, (10) comparison of the proposed model with existing work in terms of energy saving, and (11) effect of energy-aware resource management on network utilization.

7.1. Effect of Route Number on Load Balancing Performance

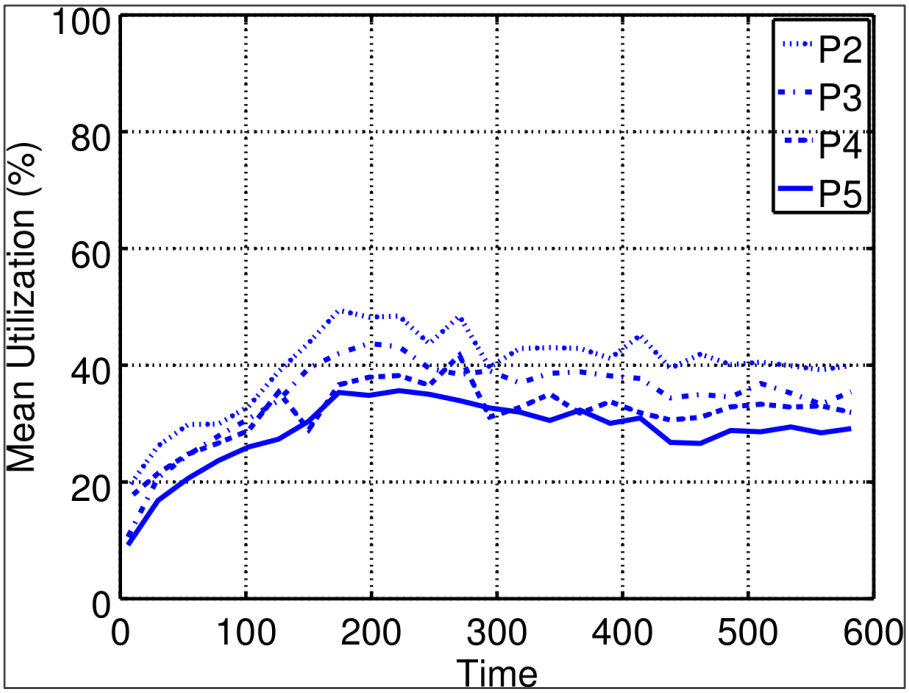

Figure 7.5: The core link utilizations in time for different number of routes.

Load balancing performance of the proposed framework depends on the number of routes. In this context, we have investigated the performance of load balancing under various route numbers, namely P2, P3, P4, and P5. P2, P3, P4, and P5 have 2, 3, 4, and 5 routes per pair, respectively.

Load balancing performance of the proposed framework depends on the number of routes. In this context, we have investigated the performance of load balancing under various route numbers, namely P2, P3, P4, and P5. P2, P3, P4, and P5 have 2, 3, 4, and 5 routes per pair, respectively. Net-M is chosen as the test topology since it allows creation of 5 disjoint routes, unlike others. The same traffic trace combination is applied to the same pairs of Net-M for 10 minutes. There are three test runs. During each test run, maximum core link utilizations are measured and the average of them is calculated per measurement time. The result is illustrated in Figure 7.5.

We have observed that performing load balancing with 2 routes has the worst performance as it has the highest utilization. In the P2 case, the controller performs load balancing among two routes per pair. As the proposed framework performs

adaptive load balancing, the convergence of cost equalization among pairs takes more time. As the route number increases, max utilization decreases. For the same reason, P5 has better performance since more pairs are neighbors.

7.2. Effect of Periodic Execution on Load Balancing Performance

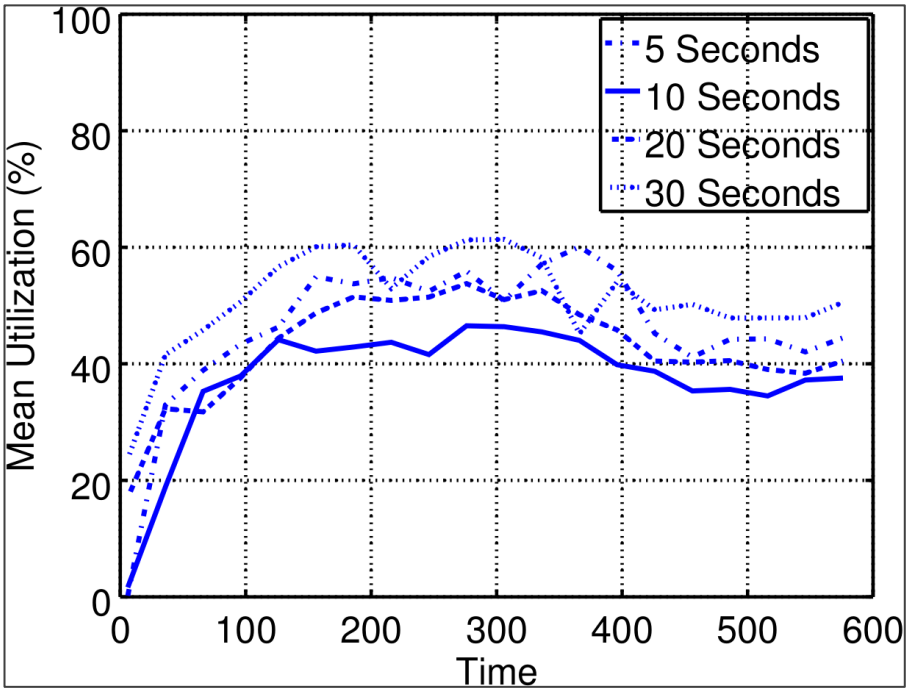

Figure 7.6: The core link utilizations of P2 in time for various load balancing periods.

We have investigated the performance of load balancing under various periods to understand how frequently the load balancing should be performed depending on the route number. Load balancing periods are set to 5, 10, 20, and 30 seconds. Net-M is chosen as the test topology since it allows to creation of 5 disjoint routes, unlike others. The same traffic trace combination is applied to the same pairs of Net-M for 10 minutes. There are three test runs for both P2 and P5. During each test run, maximum core link utilizations are measured and the average of them is calculated per measurement time. The results are illustrated in Figure 7.6 and Figure 7.7.

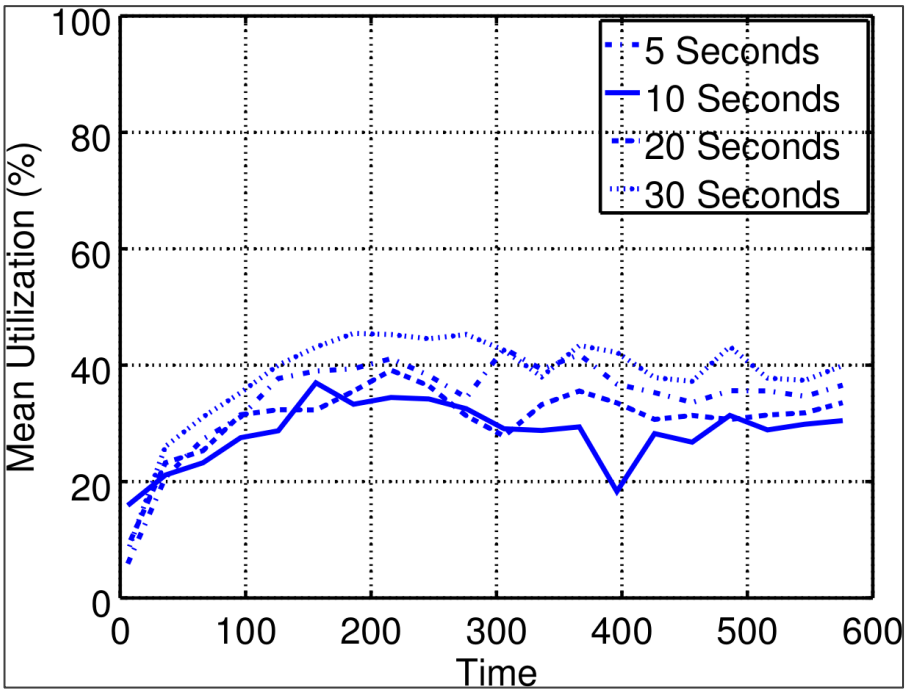

Figure 7.7: The core link utilizations of P5 in time for various load balancing periods.

We have observed that performing load balancing in short (e.g., 5 seconds) or large (e.g., 30 seconds) intervals both for P2 and P5 causes performance degradation. Performing load balancing with short period shifts flows from one route to another frequently. This causes unnatural fluctuation in traffic. Load balancing with a large period has low performance as it equals costs infrequently. This implies that load balancing with larger periods makes the case like there is no load balancing, especially for highly dynamic network traffic.

7.3. Effect of Threshold-based Execution on Load Balancing Performance

The proposed load-balancing method can also be initiated depending on the cost difference among routes of pairs. In this context, we have investigated the performance of load balancing under various cost differences, in other words, threshold-based load balancing (TLB), which are 0.1, 0.25, 0.5, and 1.0. Net-M is chosen as the test topology

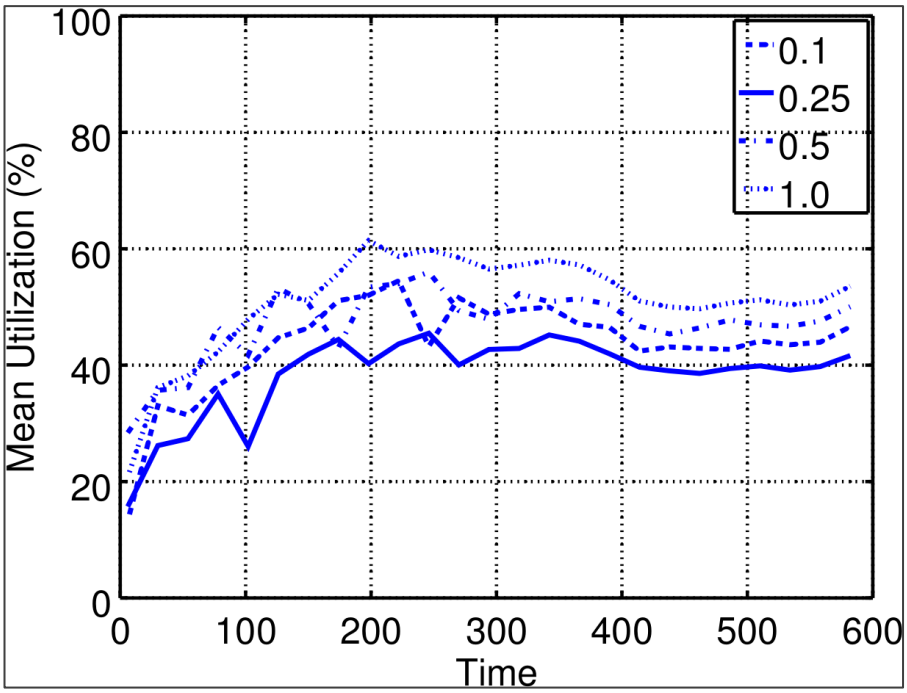

Figure 7.8: The core link utilizations of P2 in time for various load balancing periods.

since it allows to creation of 5 disjoint routes. The same traffic trace combination is applied to the same pairs of Net-M for 10 minutes. There are three test runs for both P2 and P5. During each test run, maximum core link utilizations are measured and the average of them is calculated per measurement time. The results are illustrated in Figure 7.8 and Figure 7.9.

Similar to the periodic load balancing case, we have observed that large TLBs (e.g., 1.0) has little impact on cost equalization. TLB with 1.0 has the lowest performance as the controller has initiated the load balancing process a few times. Additionally, performing load balancing for very small cost differences (e.g., 0.1) causes lower utilization compared to TLB with 0.25 and 0.5. This is because the controller frequently performs flow shifting among routes of pairs. We have observed that TLB with 0.25 has the best performance regardless of route numbers among the threshold values we chose. Similar to the periodic load balancing, the proposed model achieves better utilization with a larger number of routes. It must be noted that TLB should be adjusted based on the traffic fluctuation.

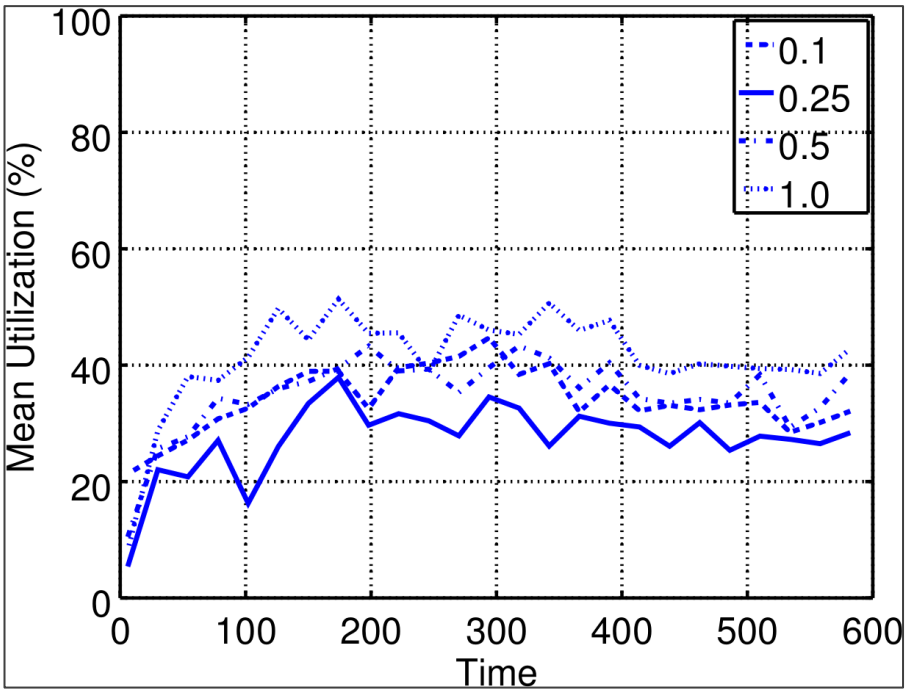

Figure 7.9: The core link utilizations of P2 in time for various load balancing periods.

7.4. Load Balancing Performance under Various Traffic Loads

We have investigated how the proposed framework performs under various traffic loads (eg., 20% - light, 50% - moderate, 80% - heavy). To do this, the same set of traffic traces are applied on Net-M with two routes per pair by scaling them. There are three test runs. The duration of the test run is set to 10 minutes and periodic (i.e., 10 seconds) load balancing is performed by the controller. During each test run per traffic load, maximum core link utilizations are measured and the average of them is calculated per measurement time. The result is illustrated in Figure 7.10.

To generate a light traffic load flows with small sizes are generated by clients. In case of the moderate traffic load generation, there is a mixture of flows with small and large sizes. In the heavy traffic load case, almost all flows are large. Under these

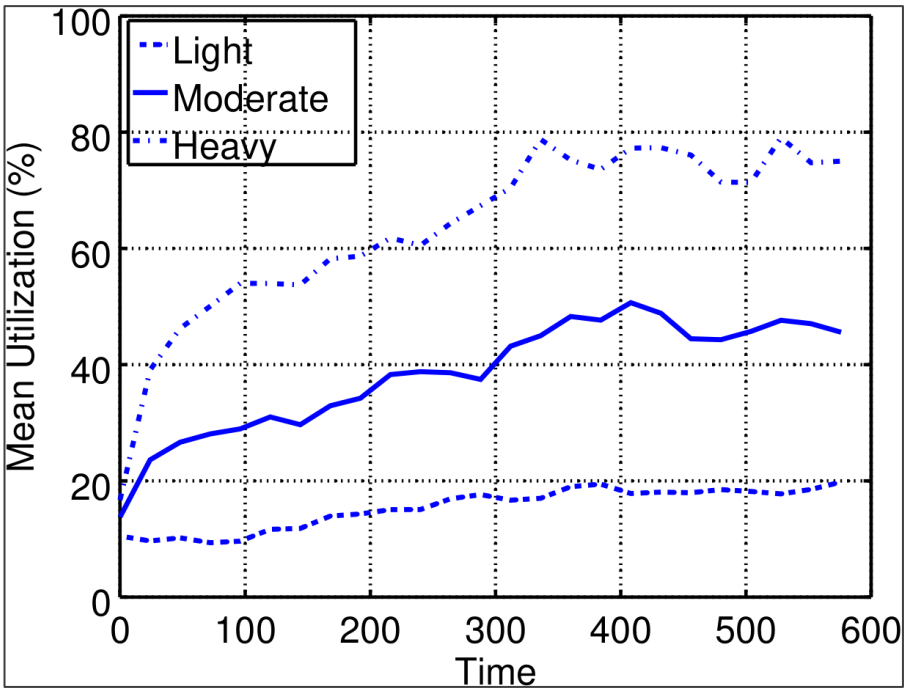

Figure 7.10: The core link utilizations for various traffic loads.

circumstances, we have observed that shifting flows from one route to another causes small cost changes in light traffic load case. Fluctuation in the moderate traffic load case is more compared to the previous case. In heavy traffic load case, the impact of shifting flows from one route to another causes greater fluctuation. In the real world, there are many flows of various sizes. Hence, load balancing can be achieved more smoothly.

7.5. Load Balancing Performance for Various Topologies

We have investigated the performance of the proposed model for different networks. In this context, we applied the same set of traffic traces to all test topologies for 10 minutes. The applied traffic load is moderate. We have measured the change in costs of two randomly chosen routes. The obtained results, illustrated in Figure 7.11, Figure 7.12, and Figure 7.13, show that route costs get close to each other after the cost equalization process. Therefore, the proposed model successfully balances loads regardless of topology in an adaptive manner.

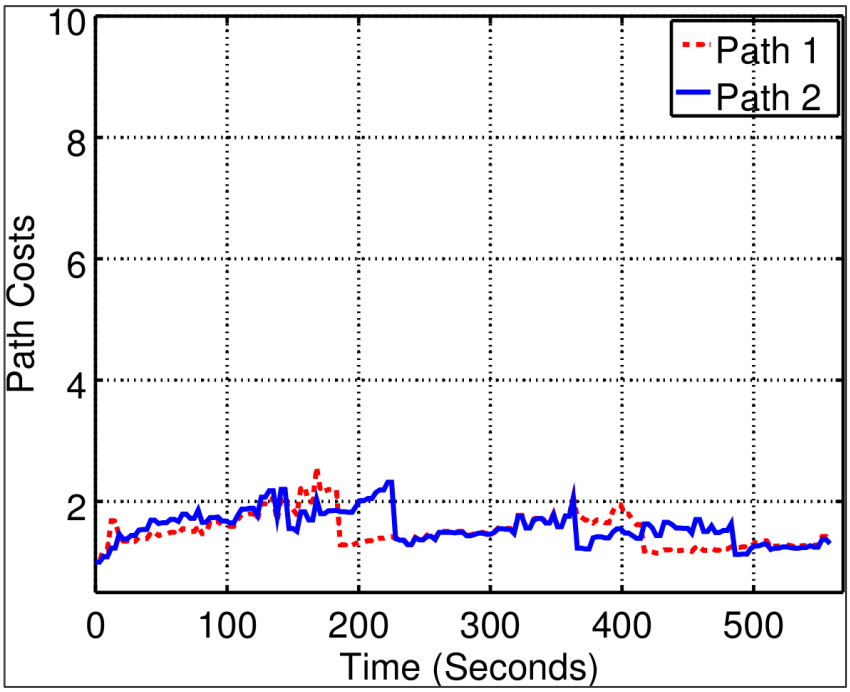

Figure 7.11: Equalization of route costs for NSFNET under moderate traffic load.

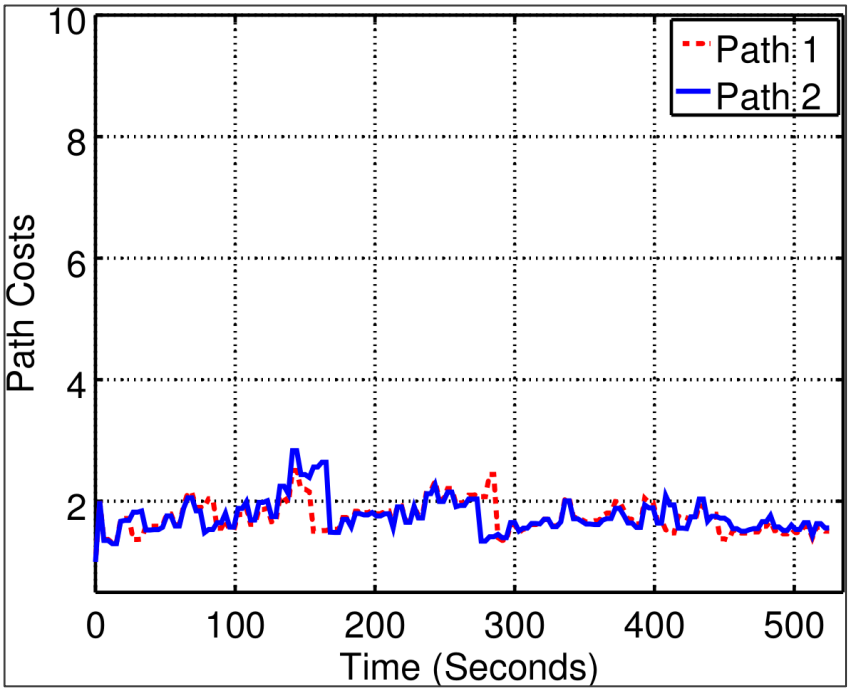

Figure 7.12: Equalization of route costs for Net-M under moderate traffic load.

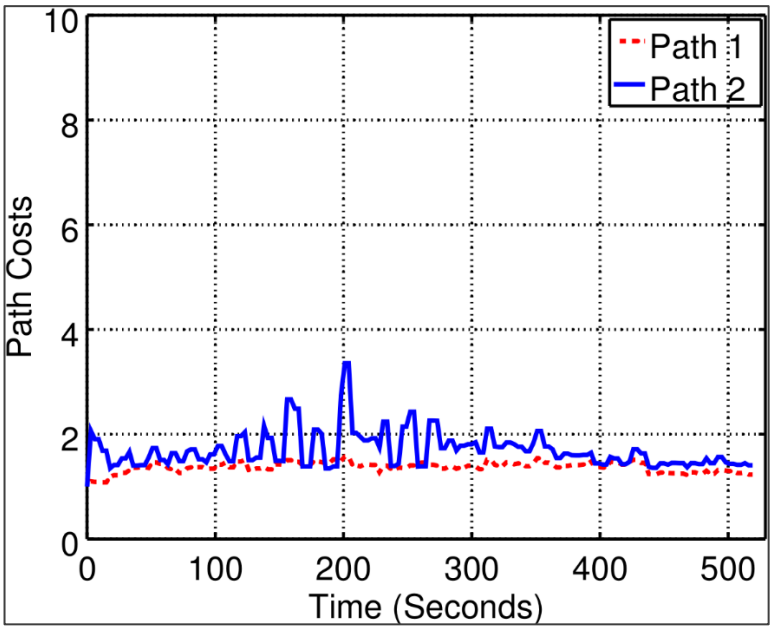

Figure 7.13: Equalization of route costs for Net-L under moderate traffic load.

7.6. Load Balancing Performance Comparison

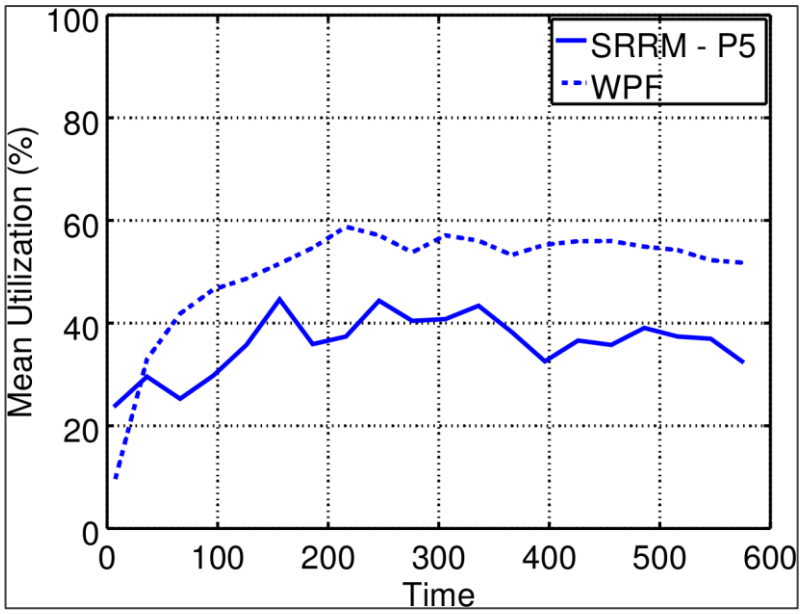

Figure 7.14: Comparison of load balancing performance of models.

The works [51] and [52] address both resource management and control plane scalability in SDN-based networks. However, they are complex to implement due to their both hierarchical and distributed structures. They also contain multiple controllers. Therefore, instead of comparing our model with the proposed works in

[51] and [52], we compare our model with the WPF, which is implemented in [87]. WPF also has a single controller that performs both resource utilization and routing. In this test scenario, we applied the same set of traffic traces to both models for 10 minutes. Net-M is chosen as the test topology. The applied traffic load is moderate. There are three test runs. During each test run, maximum core link utilizations are measured and the average of them is calculated per measurement time. The result is illustrated in Figure 7.14. We have investigated performance.

Note that WPF calculates the shortest route in a graph where both links and nodes have had weighted. WPF establishes a route once for a flow and does not change the route in time. Thus, link utilization does not change drastically in time. However, the proposed model, where each pair has 5 routes, successfully performs resource utilization and outperforms WPF.

7.7. Admission Control Performance

In this test scenario, we have investigated the performance of the admission control mechanism implemented in the proposed model. In the first part of this test case, we have measured the admission control time for both SRRM and WPF and illustrated the results in Figure 7.15. We have observed that WPF consumes more time to find a suitable route as the network load increases. The proposed model, on the other hand, performs admission control very quickly. Besides, the increase in admission control time for increasing traffic load is negligible. This proves that our model performs admission control in a scalable manner.

We have conducted another test on Net-M to examine the effect of resource management operations on flow acceptance rate. We sent periodic flow requests to only one arbitrarily chosen pair and measured the rate of flow acceptance and run this scenario

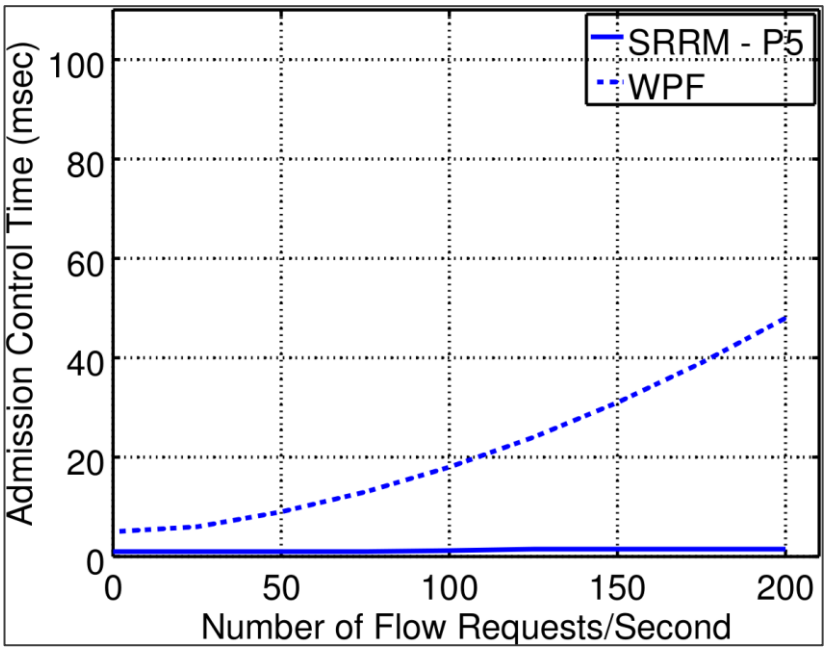

Figure 7.15: Comparison of admission control time performance of models.

both for P2 and P5. Our controller has distributed network load among routes and performed resizing to reduce congestion, if necessary. Thus, flow acceptance rates of SRRM is close to WPF as illustrated in Figure 7.16 even P2 case.

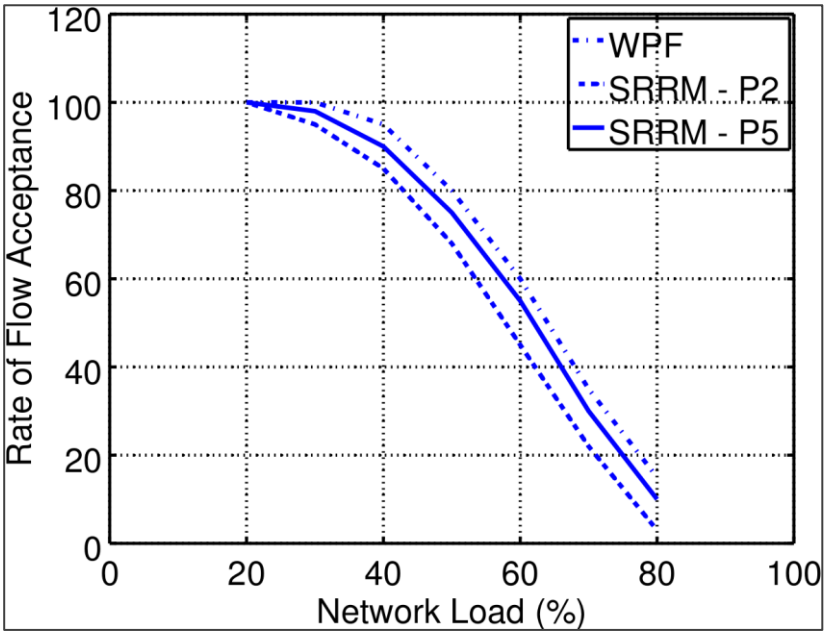

Figure 7.16: Comparison of flow acceptance rate performance of models.

7.8. Effects of Route Number per Pair and Traffic Load on Energy Saving Performance

In this test scenario, we have investigated the energy-saving performance of the proposed model for a various number of routes per pair under light, moderate, and heavy traffic loads. Route number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. There are three test runs. During each run, active link and switch ratios are recorded and the average of each run is calculated. Figure 7.17, Figure 7.18, Figure 7.19, and Figure 7.20 illustrate the energy-saving performance of P2 and P5 in terms of link and switch.

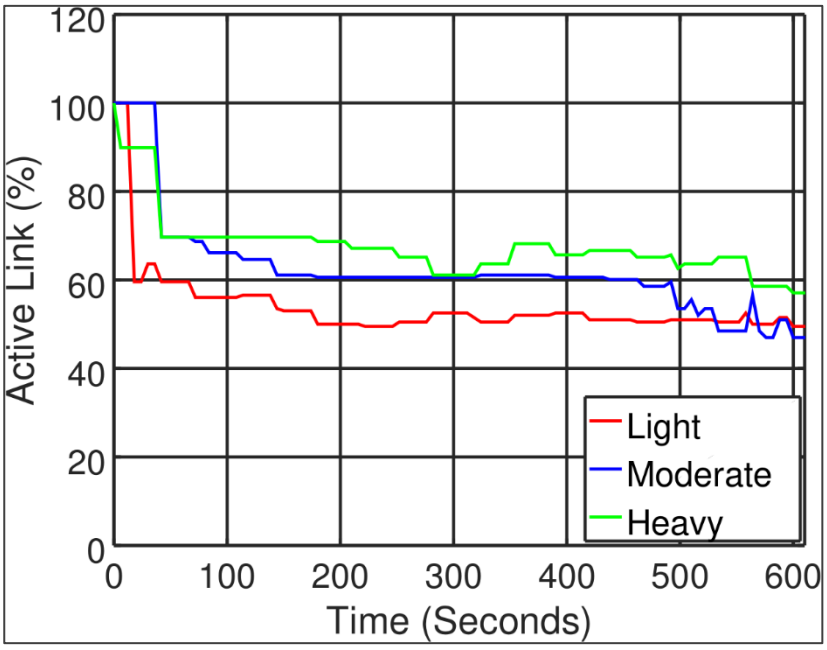

Figure 7.17: Active link ratios for P2 under various traffic loads.

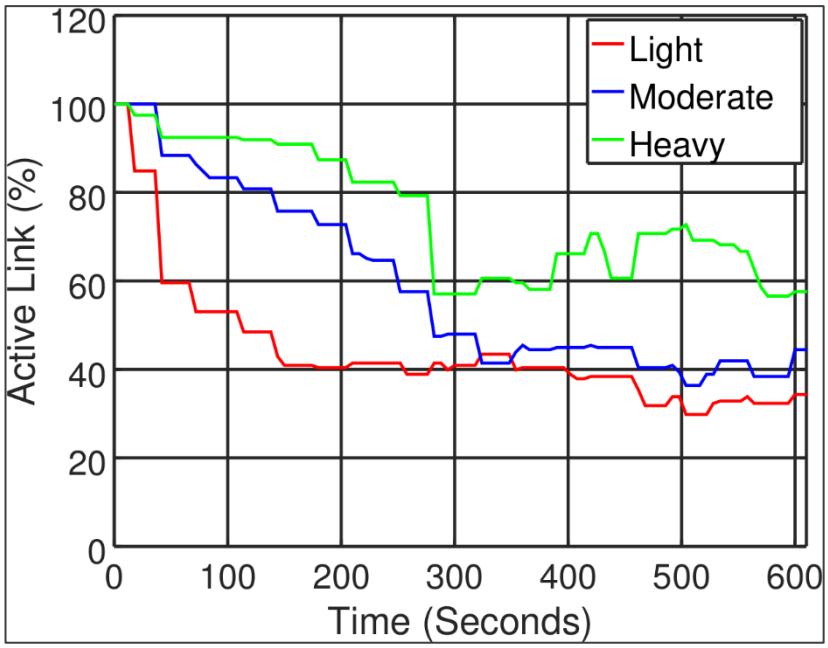

Figure 7.18: Active link ratios for P5 under various traffic loads.

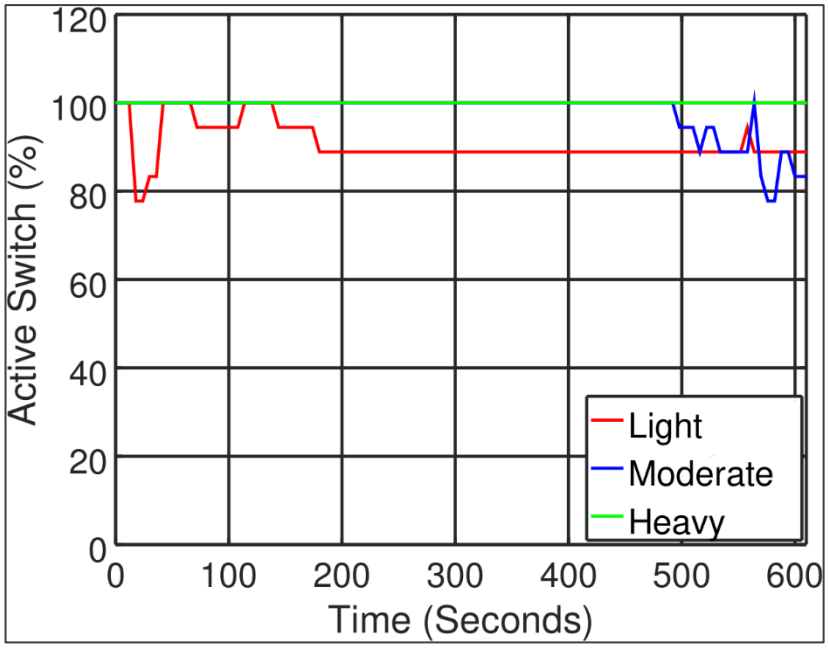

Figure 7.19: Active switch ratios for P2 under various traffic loads.

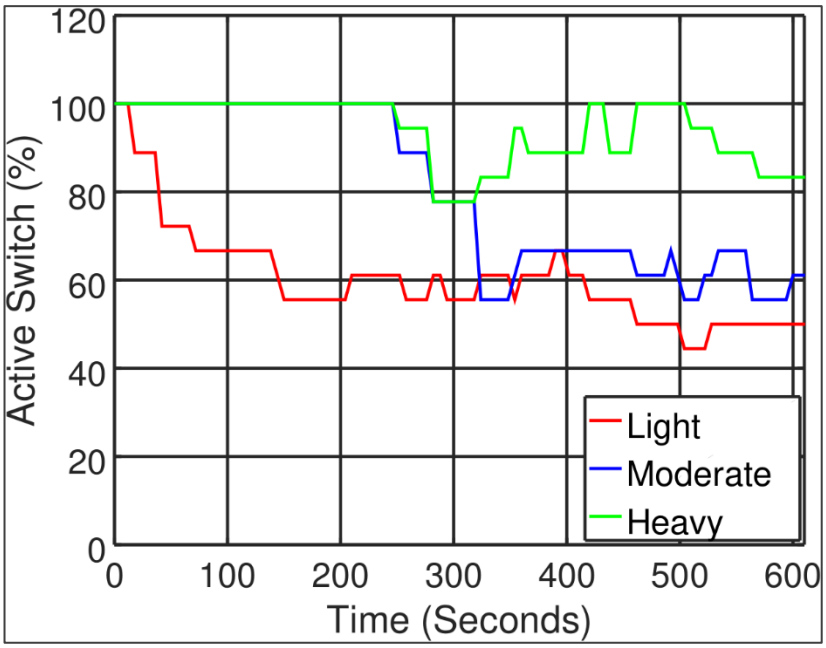

Figure 7.20: Active switch ratios for P5 under various traffic loads.

We have observed that the active link ratio drops down to around 50-60% in P2 for all traffic loads as illustrated in Figure 7.17. This experiment results in lower power consumption in P5 for light and moderate traffic loads as illustrated in Figure 7.18 since the active link ratio drops down to around 35-45% at the end of the experiment. Note that, the energy-saving performance of both P2 and P5 is the closest to each other under heavy traffic load because the controller needs more resources to carry the heavy load. Also, notice that the active link ratio of P2 decreases to around 60-70% in almost 50 seconds compared to P5. This ratio does not change much since then. The reason is that the controller has less choice in terms of route number per pair. However, the energy-saving performance of P5 is better after 300 seconds. The difference between the active switch ratio of P2 and P5 is much more apparent as shown in Figure 7.19, and Figure 7.20 for all traffic loads. P2 shows poorer performance even under light traffic load. In case of heavy traffic load, the controller fails to make any switch passive. However, this is not the case for P5. The active switch ratio of P5 drops down to around 60% for light and moderate traffic loads. Although its energy-saving performance under heavy traffic load is not as good as others, the controller successfully makes a few amounts of the switches passive.

7.9. Effect of Connectivity on Energy Saving Performance

In this test scenario, we have investigated the energy-saving performance of the proposed model for different topologies. The average degree of connectivity for NSFNET, Net-M, and Net-L are 3, 6, and 3.53, respectively. Route number per pair is 2 and a moderate traffic load is applied. There are three test runs. During each run, active link and switch ratios are recorded and the average of each run is calculated.

We have observed that the degree of connectivity does not affect the energy-saving performance of the proposed model on links as illustrated in Figure 7.21, and Figure 7.22. The active link ratio in all topologies is close to each other. On the other hand, this is not

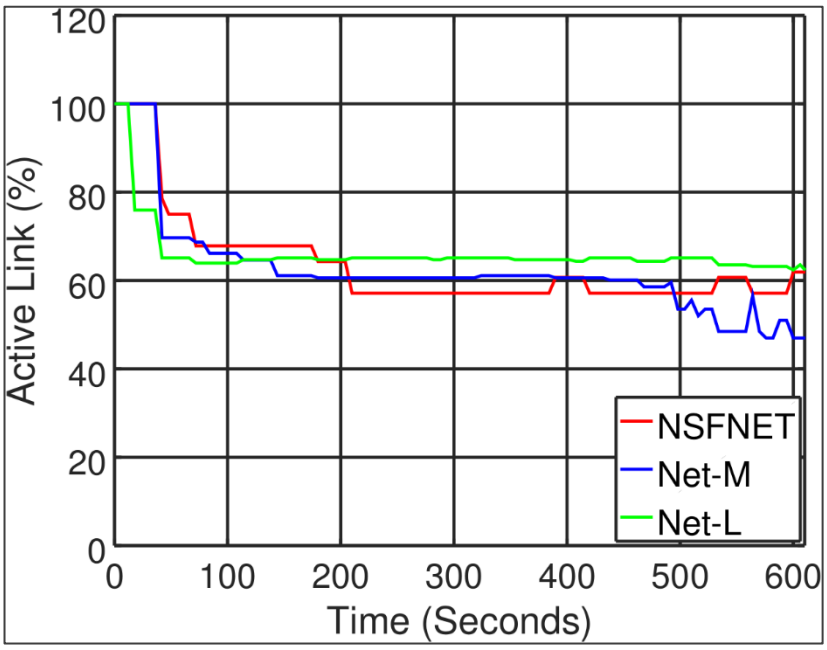

Figure 7.21: Active link ratios of 3 topologies having 2 routes per pair under moderate traffic load.

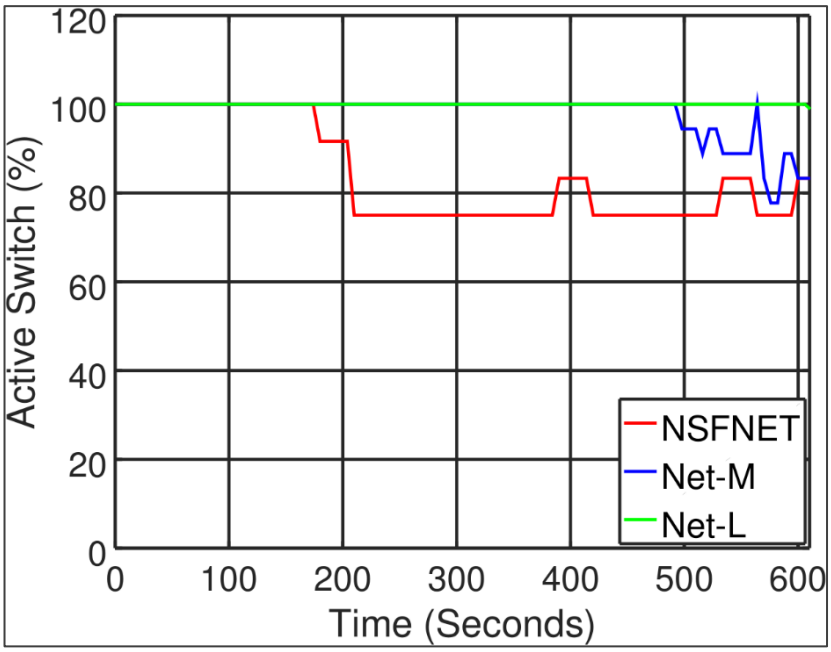

Figure 7.22: Active switch ratios of 3 topologies having 2 routes per pair under moderate traffic load.

the case for switches. NSFNET has the best performance among other topologies because routes in this topology have more common links and switches. Net-L has the worst performance among others because routes in Net-L are more diverse than others due to the high connectivity. This test case shows that the establishment of routes affects the energy-saving performance of the proposed model.

7.10. Comparison of Energy Saving Performance

In this test case, we have investigated the energy-saving performance of the proposed model by comparing it with the approach defined in [59]. In our model, as aforementioned, we define route energy consumption as the multiplication of PNR and PDV. In [59], authors define energy consumption based on link utilization and a congestion threshold, that is 80%. We call this model the CARE model. In this test case, we run ERMA both for the proposed method and CARE model. Route number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. Moderate traffic is applied in all 3 test runs. During each run, active link and switch ratios are recorded and the average of each run is calculated. Figure 7.23, Figure 7.24, Figure 7.25, and Figure 7.26 illustrate energy-saving performance of the proposed and CARE models in terms of link and switch.

On one hand, the two models have almost the same performance for P2 in terms of active link ratio case as depicted in Figure 7.23. On the other hand, the proposed model outperforms the CARE model in the P5 case as shown in Figure 7.24. Similarly, the results of the active switch ratio test resemble active link ratio test. The two models have almost the same performance for P2 as illustrated in Figure 7.25, and the proposed model outperforms the CARE model for P5 as depicted in Figure 7.26.

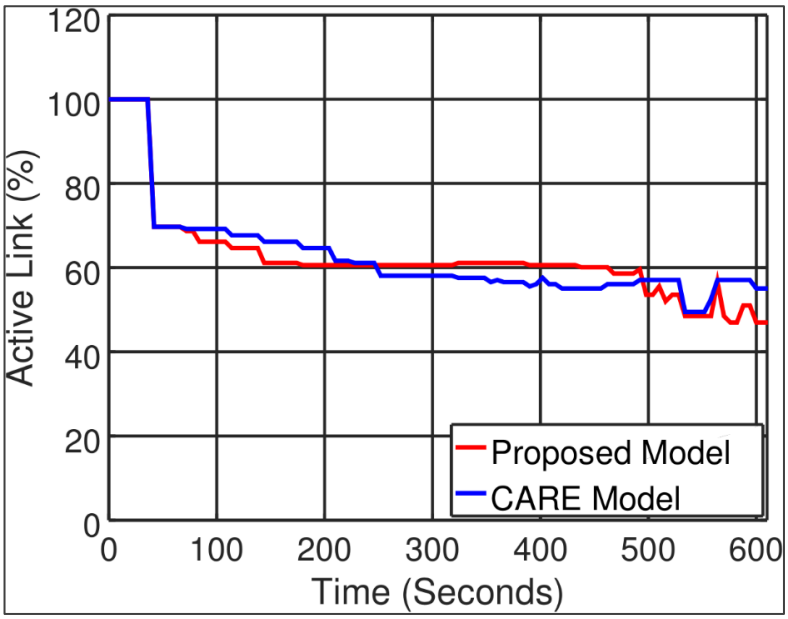

Figure 7.23: Comparison of energy saving performance based on active link ratio for P2.

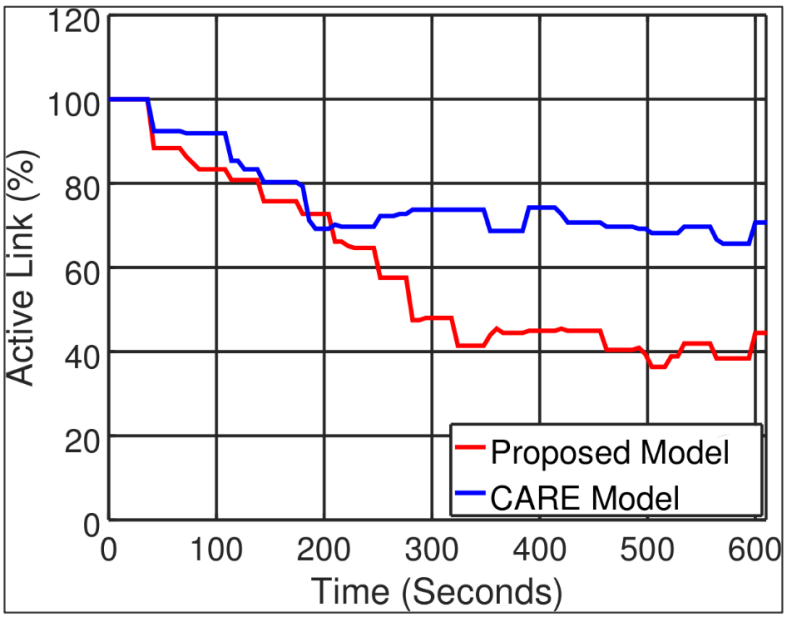

Figure 7.24: Comparison of energy saving performance based on active link ratio for P5.

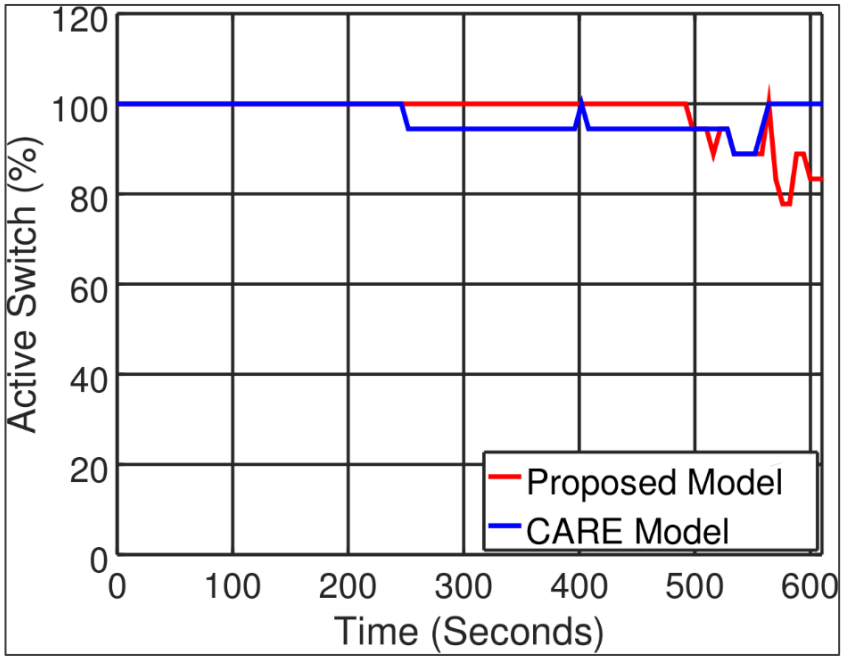

Figure 7.25: Comparison of energy saving performance based on active switch ratio for P2.

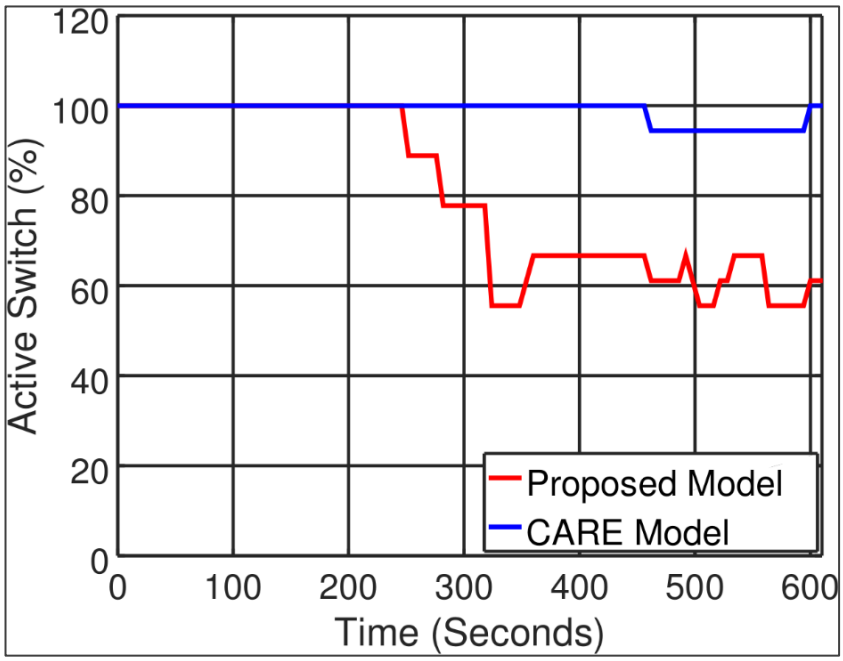

Figure 7.26: Comparison of energy saving performance based on active switch ratio for P5.

7.11. Comparison of Load Balancing Performance

In this test case, we have compared the load balancing performance when the controller performs energy saving and load balancing in harmony (Energy-Aware: EA) or it only performs load balancing (Energy-Unaware: EU). In EA, load balancing is performed among active routes. In EU, routes are always active. Route number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. Moderate traffic is applied in all 3 test runs. During each run, load rates are recorded, and the average of each run is calculated. Figure 7.27, and Figure 7.28 illustrate load balancing performance of both models for P2 and P5.

In both P2 and P5, the utilization of links is higher in EA compared to EU. This is because the same load is distributed over less capacity due to passive routes in EA. Additionally, utilization increases in line with energy-saving. In previous test cases, we have observed that energy-saving performance increases conforming to route number per pair. Hence, load utilization of P2 as in Figure 7.27 is lower than P5 as in Figure 7.28. The trade-off between energy saving and load balancing indicates that high energy saving may cause over-utilization. Thus, this may result in significant network congestion. Finally, this test case shows that an adequate route number per pair and PDV selection allows efficient usage of network resources in terms of both energy saving and resource utilization.

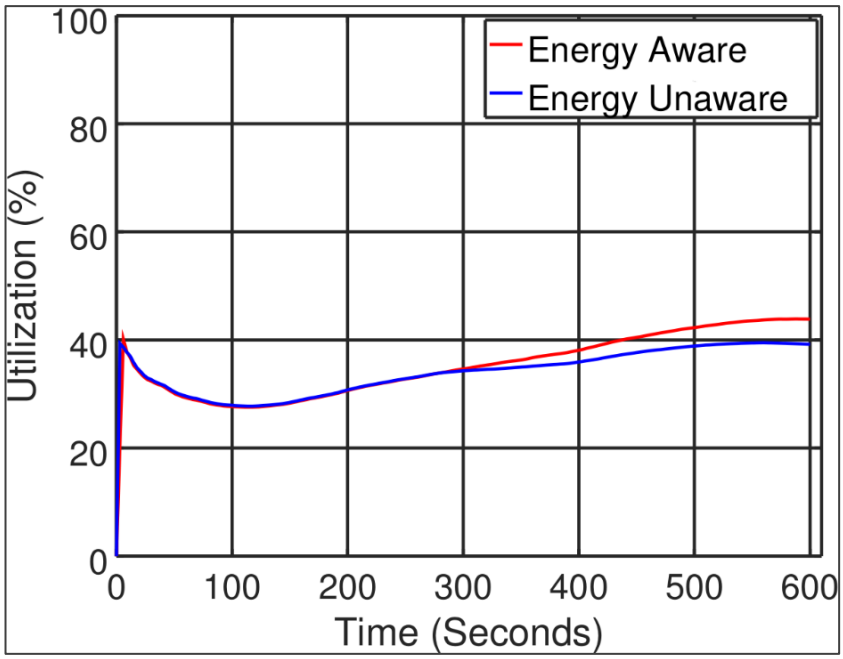

Figure 7.27: Comparison of load balancing performance for P2.

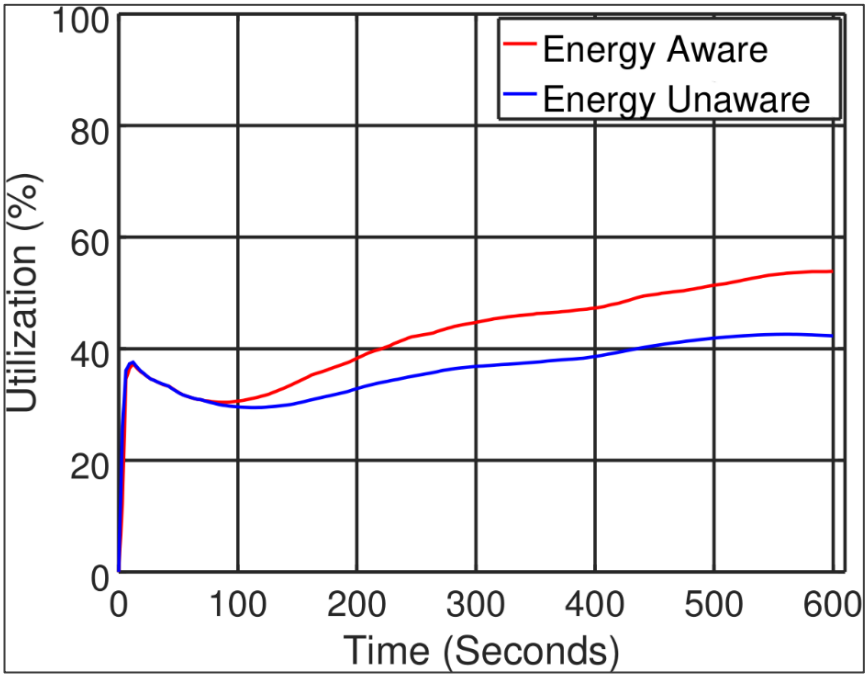

Figure 7.28: Comparison of load balancing performance for P5.

8. CONCLUSION

To put it in a nutshell, we propose a scalable routing and energy-aware resource management model for SDN-based SPNs. In the proposed model, routes are pre-computed between each pair of edge switches. These routes convert a complex physical network to a simple and virtual network. The controller performs scalable routing, admission control, and signaling based on these routes. It also performs energy-aware resource management. The controller saves energy via aggregating flows to a smaller number of routes per edge pair and deactivating unused routes. During this process, it takes neighbor routes and load level into account. Pair-based power-saving converges to global power-saving in time. In line with this, the controller also performs adaptive load balancing by equalizing active route costs per pair. In the case of high network load, the controller adjusts route capacities for further resource utilization.

The proposed model either accepts or rejects flow requests upon receiving requests due to its pre-computed route structure. Instead of computing flow routes per request, routes that are established in off-line mode prevent the controller to become a bottleneck. Thus, the proposed model shows that even if there is a single controller at the top of the network, it can successfully perform routing, and admission control in a scalable manner. We also witness that the amount of signaling between controller and data plane is also reduced both in statistic collection and route establishment processes. In addition to these, our model can save energy and balances the pair load among its routes successfully. Moreover, we observe that route resizing is a heavy process, but the controller performs this operation rarely. Therefore, it does not affect the scalability of the controller significantly. Finally, obtained results prove that load balancing and route resizing increase the flow acceptance rate dramatically.

As future work, controller can perform load balancing by estimating incoming traffic. Also, the controller can utilize a hybrid version of periodic and threshold-based load balancing. In this version, the controller can estimate the time to perform load balancing. Apart from these, the controller can increase the energy-saving performance of the proposed model using deep learning techniques.

REFERENCES

[1] Web 1, (2015), https://blogs.cisco.com/sp/the-history-and-future-of-internettraffic, (Accessed: 15/01/2021).

[2] Web 2, (2019), https://www.itu.int/ en/ITU-D/Statistics/Documents/facts/Facts Figures2019.pdf, (Accessed: 15/01/20 21).

[3] Web 3, (2020), https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/ Downloads/2020/Phenomena/COVID_Internet_Phenomena_Report_20200507 .pdf, (Accessed: 15/01/2021).

[4] Chabarek J., Sommers J., Barford P., Estan C., Tsiang D., Wright S., (2008), "Power Awareness in Network Design and Routing", The 27th Conference on Computer Communications IEEE INFOCOM, 457-465, Phoenix, AZ, 13-18 April.

[5] Awad M. K., Neama G., Rafique Y., (2015), "The impact of practical network constraints on the performance of energy-aware routing schemes", 2015 IEEE International Conference on Service Operations And Logistics and Informatics (SOLI), 77–81, Hammamet, Tunisia, 15-17 November.

[6] Tuysuz M. F., Ankarali Z. K., Gözüpek D., (2017), "A survey on energy efficiency in software defined networks", Computer Networks, 113 (C), 188–204.

[7] Van Heddeghem W., Lambert S., Lannoo B., Colle D., Pickavet M., Demeester P., (2014), "Trends in Worldwide ICT Electricity Consumption from 2007 to 2012", Computer Communications, 50, 64–76.

[8] Web 4, (2009), https://www.cs.odu.edu/~cs752/papers/sdr-infocom_brazil_2009_ v1-1.pdf, (Accessed: 15/01/2021).

[9] Yeganeh S. H., Tootoonchian A., Ganjali Y., (2013), "On scalability of software-defined networking", IEEE Communications Magazine, 51 (2), 136-141.

[10] Luo H., Cui J., Chen G., Chen Z., Zhang H., (2014), "On the applicability of software defined networking to large scale networks", 23rd International Conference on Computer Communication and Networks (ICCCN), 1–6, Shanghai, China, 4-7 August.

[11] Leguay J., Maggi L., Draief M., Paris S., Chouvardas S., (2016), "Admission control with online algorithms in SDN", IEEE/IFIP Network Operations and Management Symposium (NOMS), 718–721, Istanbul, Turkey, 25-29 April.

[12] Bianco A., Giaccone P., Mahmood A., Ullio M., Vercellone V., (2015),

"Evaluating the SDN control traffic in large ISP networks", IEEE International Conference on Communications (ICC), 5248–5253, London, UK, 8-12 June.

[13] McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J., (2008), "OpenFlow: Enabling Innovation in Campus Networks", ACM SIGCOMM Computer Communication Review, 38 (2), 69-74.

[14] Berde P., Gerola M., Hart J., Higuchi Y., Kobayashi M., Koide T., Lantz B., O'Connor B., Radoslavov P., Snow W., Parulkar G., (2014), "ONOS: Towards an Open, Distributed SDN OS", Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, 1–6.

[15] Medved J., Varga R., Tkacik A., Gray K., (2014), "OpenDaylight: Towards a model-driven SDN controller architecture", IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 1–6, Sydney, NSW, Australia, 19 June.

[16] Web 5, (2014), https://floodlight.atlassian.net/wiki/spaces/ floodlightcontroller/ pages/1343555/Floodlight+v1.0, (Accessed: 15/01/2021).

[17] Foster N., Harrison R., Freedman M. J., Monsanto C., Rexford J., Story A., Walker D., (2011), "Frenetic: A Network Programming Language", SIGPLAN Notices, 46 (9), 279–291.

[18] Otoshi T., Ohsita Y., Murata M., Takahashi Y., Ishibashi K., Shiomoto K., (2013), "Traffic prediction for dynamic traffic engineering considering traffic variation", IEEE Global Communications Conference (GLOBECOM), 1570–1576, Atlanta, GA, USA, 9-13 December.

[19] Monsanto C., Reich J., Foster N., Rexford J., Walker D., (2013), "Composing Software Defined Networks", 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 1–13, Lombard, IL, USA, 2-5 April.

[20] Web 6, (2009), https://www.opennetworking.org/images/stories/downloads /sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, (Accessed: 15/ 01/2021).

[21] Web 7, (2010), http://www.rfc-editor.org/rfc/rfc5810.txt, (Accessed: 15/01/2021).

[22] Web 8, (2011), https://rfc-editor.org/rfc/ rfc6241.txt, (Accessed: 15/01/2021).

[23] Ongaro D., Ousterhout J., (2014), "In Search of an Understandable Consensus Algorithm", USENIX Annual Technical Conference, 305–320, Philadelphia, PA, USA, 19-20 June.

[24] Muqaddas A. S., Bianco A., Giaccone P., Maier G., (2016), "Inter-controller traffic in ONOS clusters for SDN networks", IEEE International Conference on

Communications (ICC), 1–6, Kuala Lumpur, Malaysia, 22-27 May.

[25] Wyatt D., (2013), "Akka Concurrency: Building Reliable Software in a Multi-Core World", 1st Edition, Artima Incorporation.

[26] Zhang T., Giaccone P., Bianco A., De Domenico S., (2017), "The Role of the Inter-Controller Consensus in the Placement of Distributed SDN Controllers", Computer Communications, 113 (C), 1-13.

[27] Web 9, (2012), https://datatracker.ietf.org/doc/html/draft-yin-sdn-sdni-00, (Accessed: 15/01/2021).

[28] Akyildiz I. F., Lee A., Wang P., Luo M., Chou W., (2016), "Research challenges for traffic engineering in software defined networks", IEEE Network, 30 (3), 52-58.

[29] Tootoonchian A., Ganjali Y., (2010), "HyperFlow: A Distributed Control Plane for OpenFlow", Internet Network Management Conference on Research on Enterprise Networking, 1-3, San Jose, CA, USA, 28-30 April.

[30] Koponen T., Casado M., Gude N., Stribling J., Poutievski L., Zhu M., Ramanathan R., Iwata Y., Inoue H., Hama T., Shenker S., (2010), "Onix: A Distributed Control Platform for Large-scale Production Networks", 9th USENIX Conference on Operating Systems Design and Implementation, 351–364, Vancouver, BC, Canada, 4-6 October.

[31] Agarwal S., Kodialam M., Lakshman T. V, (2013), "Traffic engineering in software defined networks", IEEE INFOCOM, 2211–2219, Turin, Italy, 14-19 April.

[32] Gvozdiev N., Karp B., Handley M., (2014), "FUBAR: Flow Utility Based Routing", 13th ACM Workshop on Hot Topics in Networks, 1–7, Los Angeles, CA, USA, 27-28 October.

[33] Vissicchio S., Vanbever L., Rexford J., (2014), "Sweet Little Lies: Fake Topologies for Flexible Routing", Proceedings of the 13th ACM Workshop on Hot Topics in Networks, 1–7, Los Angeles, CA, USA, 27-28 October.

[34] Kohler T., Dürr F., Rothermel K., (2017), "ZeroSDN: A Highly Flexible and Modular Architecture for Full-Range Network Control Distribution", ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 25–37, Beijing, China, 18-19 May.

[35] Lin P., Bi J., Hu H., (2012), "ASIC: An Architecture for Scalable Intra-Domain Control in OpenFlow", 7th International Conference on Future Internet Technologies, 21–26, Seoul, Korea, 11-12 September.

[36] Hassas Yeganeh S., Ganjali Y., (2012), "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications", First Workshop on Hot Topics in Software Defined Networks, 19–24, Helsinki Finland, 13 August.

[37] Ahmed R., Boutaba R., (2014), "Design considerations for managing wide area software defined networks", IEEE Communications Magazine, 52 (7), 116-123.

[38] Lopez-Rodriguez F., Campelo D. R., (2014), "A robust SDN network architecture for service providers", IEEE Global Communications Conference (GLOBECOM), 1903–1908, Austin, TX, USA, 8-12 December.

[39] Wang J., Shou G., Hu Y., Guo Z., (2016), "A Multi-Domain SDN Scalability Architecture Implementation Based on the Coordinate Controller", International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 494–499, Chengdu, Sichuan, China, 13-15 October.

[40] Yu M., Rexford J., Freedman M. J., Wang J., (2010), "Scalable flow-based networking with DIFANE", SIGCOM Computer Communication Review, 40 (4), 351-362.

[41] Hu Y., Wang W., Gong X., Que X., Cheng S., (2012), "BalanceFlow: Controller load balancing for OpenFlow networks", 2nd IEEE International Conference on Cloud Computing and Intelligence Systems, 780–785, Hangzhou, China, 30 October-1 November.

[42] Othman M. M. O., Okamura K., (2013), "Enhancing Control Model to Ease Off Centralized Control of Flow-Based SDNs", 37th Annual IEEE Computer Software and Applications Conference, 467–470, Kyoto, Japan, 22-26 July.

[43] Jain S., Kumar A., Mandal S., Ong J., Poutievski L., Singh A., Venkata S., Wanderer J., Zhou J., Zhu M., Zolla J., Hölzle U., Stuart S., Vahdat A., (2013), "B4: Experience with a Globally-deployed Software Defined Wan", SIGCOMM Computer Communications Review, 43 (4), 3-14.

[44] Wang J. M., Wang Y., Dai X., Bensaou B., (2014), "SDN-based multi-class QoS-guaranteed inter-data center traffic management", 3rd IEEE International Conference on Cloud Networking (CloudNet), 401–406, Luxembourg, 8-10 October.

[45] Stefano A. D., Cammarata G., Morana G., Zito D., (2015), "A4SDN - Adaptive Alienated Ant Algorithm for Software-Defined Networking", 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 344–350, Krakow, Poland, 4-6 November.

[46] Cammarata G., Stefano A. D., Morana G., Zito D., (2016), "Evaluating the Performance of A4SDN on Various Network Topologies", 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 801–808, Chicago, IL, USA, 23-27 May.

[47] Tomovic S., Lekic N., Radusinovic I., Gardasevic G., (2016), "A new approach to dynamic routing in SDN networks", 18th Mediterranean Electrotechnical

Conference (MELECON), 1–6, Lemesos, Cyprus, 18-20 April.

[48] Wang Z., Crowcroft J., (1996), "Routing algorithms for supporting resource reservation", Computer Networks, 52 (15), 2988-3006.

[49] Kodialam M., Lakshman T. V, (2000), "Minimum interference routing with applications to MPLS traffic engineering", Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 884–893, Tel Aviv, Israel, 26-30 March.

[50] Boutaba R., Szeto W., Iraqi Y., (2002), "DORA: Efficient Routing for MPLS Traffic Engineering", Journal of Network and Systems Management, 10 (3), 309–325.

[51] Luo M., Zeng Y., Li J., Chou W., (2015), "An adaptive multi-path computation framework for centrally controlled networks", Computer Networks, 83, 30-44.

[52] Tuncer D., Charalambides M., Clayman S., Pavlou G., (2015), "Adaptive Resource Management and Control in Software Defined Networks", IEEE Transactions on Network and Service Management, 12 (1), 18-33.

[53] Wang C., Hu B., Chen S., Li D., Liu B., (2017), "A Switch Migration-Based Decision-Making Scheme for Balancing Load in SDN", IEEE Access, 5, 4537-4544.

[54] Tucker R., Baliga J., Ayre R. W. A., Hinton K., Sorin W. V, (2008), "Energy consumption in IP networks", 34th European Conference on Optical Communication (ECOC), 1, Brussels, Belgium, 21-25 September.

[55] Heller B., Seetharaman S., Mahadevan P., Yiakoumis Y., Sharma P., Banerjee S., McKeown N., (2010), "ElasticTree: Saving Energy in Data Center Networks", 7th USENIX Conference on Networked Systems Design and Implementation, 249-264, San Jose, CA, USA, 28-30 April.

[56] Gupta M., Singh S., (2003), "Greening of the Internet", Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 19–26, Karlsruhe, Germany, 25-29 August.

[57] Zemmouri S., Vakilinia S., Cheriet M., (2016), "Let’s adapt to network change: Towards energy saving with rate adaptation in SDN", 12th International Conference on Network and Service Management (CNSM), 272–276, Montreal, QC, Canada, 31 October-4 November.

[58] Wang R., Jiang Z., Gao S., Yang W., Xia Y., Zhu M., (2014), "Energy-aware routing algorithms in Software-Defined Networks", 15th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 1–6, Sydney, NSW, Australia, 19 June.

[59] Thaenchaikun C., Jakllari G., Paillassa B., Panichpattanakul W., (2016), "Mitigate the load sharing of segment routing for SDN green traffic

engineering", 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 1–6, Phuket, Thailand, 24-27 October.

[60] Celenlioglu M. R., Goger S. B., Mantar H. A., (2011), "An SDN-based energy-aware routing model for intra-domain networks", 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 61–66, Split, Croatia, 17-19 September.

[61] Razmnoush R., Bakhshi B., (2016), "Green traffic engineering in SDN", 24th Iranian Conference on Electrical Engineering (ICEE), 693–698, Shiraz, Iran, 10-12 May.

[62] Fernández-Fernández A., Cervelló-Pastor C., Ochoa-Aday L., (2017), "A Multi-Objective Routing Strategy for QoS and Energy Awareness in Software-Defined Networks", IEEE Communications Letters, 21 (11), 2416-2419.

[63] Bolla R., Bruschi R., Davoli F., Gregorio L. Di, Donadio P., Fialho L., Collier M., Lombardo A., Recupero D. R., Szemethy T., (2013), "The Green Abstraction Layer: A Standard Power-Management Interface for Next-Generation Network Devices", IEEE Internet Computing, 17 (2), 82-86.

[64] Bruschi R., Lombardo A., Morabito G., Riccobene V., Bolla R., Davoli F., Lombardo C., (2014), "Green extension of OpenFlow", 26th International Teletraffic Congress (ITC), 1–6, Karlskrona, Sweden, 9-11 September.

[65] Awad M. K., Rafique Y., Alhadlaq S., Hassoun D., Alabdulhadi A., Thani S., (2016), "A greedy power-aware routing algorithm for software-defined networks", IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 268–273, Limassol, Cyprus, 12-14 December.

[66] Gomes R. L., Bittencourt L. F., Madeira E. R. M., Cerqueira E., Gerla M., (2016), "State-aware allocation of reliable Virtual Software Defined Networks based on bandwidth and energy", 13th IEEE Annual Consumer Communications Networking Conference (CCNC), 411–416, Las Vegas, NV, USA, 9-12 January.

[67] Hu Y., Luo T., Wang W., Deng C., (2016), "GreSDN: Toward a green software defined network", 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), 1–6, Kanazawa, Japan, 5-7 October.

[68] Rahnamay-Naeini M., Baidya S. S., Siavashi E., Ghani N., (2016), "A traffic and resource-aware energy-saving mechanism in software defined networks", 2016 International Conference on Computing, Networking and Communications (ICNC), 1–5, Kauai, HI, USA, 15-18 February.

[69] IEEE802.1Q, (1999), "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", IEEE.

[70] MPLS, (2001), "Multiprotocol Label Switching Architecture", "Multiprotocol

Label Switching Architecture", IETF.

[71] SEGROU, (2018), "Segment Routing Architecture", IETF.

[72] Eppstein D., (1999), "Finding the K Shortest Paths", SIAM J. Comput., 28 (2), 652–673.

[73] Abe J. O., Mantar H. A., Yayimli A. G., (2015), "k-Maximally Disjoint Path Routing Algorithms for SDN", IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 499–508, Xi'an, China, 17-19 September.

[74] Fleszar K., Mnich M., Spoerhase J., (2016), "New Algorithms for Maximum Disjoint Paths Based on Tree-Likeness", 24th Annual European Symposium on Algorithms (ESA), 433–461, Aarhus, Denmark, 22-24 August.

[75] Ciampa M., (2009), "CompTIA Security+ 2008 in Depth", 1st Edition, Cengage Learning PTR.

[76] SHS, (2012), "Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4", National Institute of Standards and Technology.

[77] Web 10, (2013), "OpenFlow Specifications v.1.3.3", https://www. opennetworking. org/images/stories/downloads/sdn-resources/onf-specifica tions /openflow/open flow-spec-v1.3.3.pdf, (Accessed: 15/01/2021).

[78] Lantz B., Heller B., McKeown N., (2010), "A Network in a Laptop: Rapid Prototyping for Software-defined Networks", 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 1-6, Monterey, California, USA, 20-21 October.

[79] Pfaff B., Pettit J., Koponen T., Jackson E. J., Zhou A., Rajahalme J., Gross J., Wang A., Stringer J., Shelar P., Amidon K., Casado M., (2015), "The Design and Implementation of Open vSwitch", 12th USENIX Conference on Networked Systems Design and Implementation, 117–130, Oakland, CA, USA, 4-5 May.

[80] Web 11, (2002), http://www.postel.org/tg/, (Accessed: 15/01/2021).

[81] Mills D. L., Braun H., (1987), "The NSFNET Backbone Network", SIGCOMM Computer Communication Review, 17 (5), 191–196.

[82] Claffy K. C., Braun H.-W., Polyzos G. C., (1994), "Tracking Long-term Growth of the NSFNET", Communications of the ACM, 37 (8), 34-45.

[83] Web 12, (2015), http://www.caida.org/data/realtime/passive/?monitor=eq uinix-chicago-dirA/, (Accessed: 29/03/2015).

[84] Web 13, (2015), http://www.caida.org/data/realtime/passive/?monitor=eq uinix-chicago-dirB/, (Accessed: 29/03/2015).

[85] Web 14, (2015), http://www.caida.org/data/realtime/passive/?monitor=equinix-sanjose-dirA/, (Accessed: 29/03/2015).

[86] Web 15, (2015), http://www.caida.org/data/realtime/passive/?monitor=equinix-sanjose-dirB/, (Accessed: 29/03/2015).

[87] Jiang J. R., Huang H. W., Liao J. H., Chen S. Y., (2014), "Extending Dijkstra's shortest path algorithm for software defined networking", The 16th Asia-Pacific Network Operations and Management Symposium, 1–4, Hsinchu, Taiwan, 17-19 September.

BIOGRAPHY

Mahmud Rasih ÇELENLİOĞLU was born in Üsküdar, İstanbul in 1989. He received his B.Sc. (2011) from the Department of Computer Engineering from Işık University, Turkey, and his M.Sc. (2013) from the Department of Computer Engineering from Gebze Institute of Technology, Turkey. He is currently a Ph.D. candidate in the Department of Computer Engineering at Gebze Technical University, Turkey. He is currently the Chief of the System Software Development Team in Cornea Aero Systems, Istanbul, Turkey. He has published two international journals and seven international conference papers. His research interests include software-defined networking, virtualization, resource management, energy efficiency, computer networks, vertical handover and optimization.

APPENDICES

Appendix A: Publications Based on this Thesis

International Journal Publications

Celenlioglu M. R., Tuysuz M. F., Mantar H. A., (2018), “An SDN-based scalable routing and resource management model for service provider networks”, International Journal of Communication Systems, 31 (8), 1-22.

Celenlioglu M. R., Mantar H. A., (2021), “Energy aware adaptive resource management model for software-defined networking-based service provider networks”, IET Network, 1-13.

International Conference Publications

Celenlioglu M. R., Goger S. B., Mantar H. A., (2014), "An SDN-based energy-aware routing model for intra-domain networks", 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 61–66, Split, Croatia, 17-19 September.

Celenlioglu M. R., Mantar H. A., (2014), "A scalable routing and admission control model in SDN-based networks", ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 231-232, Marina del Rey, CA, 20-21 October.

Celenlioglu M. R., Mantar H. A., (2015), "An SDN Based Intra-Domain Routing and Resource Management Model", IEEE International Conference on Cloud Engineering, 347-352, Tempe, AZ, USA, 9-13 March.

Celenlioglu M. R., Alsadi M., Mantar H. A., (2015), "Design, implementation and evaluation of SDN-based resource management model", 7th International Conference on New Technologies, Mobility and Security (NTMS), 1-5, Paris, France, 27-29 July.