Traffic Aware, Utility and Machine Learning Based Framework for Energy Efficiency in Software Defined Networks

by

Beakal Gizachew Assefa

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Computer Science and Engineering

August 23, 2019

Traffic Aware, Utility and Machine Learning Based Framework for Energy Efficiency in Software Defined Networks

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Beakal Gizachew Assefa

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Öznur Özkasap (Advisor)

_____

Prof. A. Murat Tekalp

_____

Asst.Prof. Didem Unat

_____

Assoc.Prof. Berk Canberk

_____

Assoc.Prof. Müge Sayıt

Date: _____

To my loving, caring and committed wife Meba Tadese, to my daughters Mihiret and Ewnet who endured the sacrifice necessary to reach this stage.

Abstract

Software Defined Networking (SDN) paradigm has the benefits of programmable network elements by separating the control and the forwarding planes, efficiency through optimized routing and flexibility in network management. As the energy costs contribute largely to the overall costs in networks, energy efficiency has become a significant design requirement for modern networking mechanisms. However, designing energy efficient solutions is non-trivial since they need to tackle the trade-off between energy efficiency and network performance. In this thesis, traffic aware, machine learning based, and end system aware SDN controller modules for energy efficiency are proposed.

As the first module of the controller, we propose a novel energy efficiency metric named Ratio for Energy Saving in SDN (RESDN) that quantifies energy efficiency based on link utility intervals. We provide integer programming formulation and method for maximizing the RESDN of the network. To the best of our knowledge, RESDN approach is unique as it measures how links are profitably utilized in terms of the amount of energy they consume with respect to their utility.

As the second module of the controller, we propose HyMER: a novel hybrid machine learning based framework for traffic aware energy efficient routing in SDN. The framework combines the advantages of supervised and reinforcement learning models. The supervised learning component consists of feature extraction, training, and testing. The reinforcement learning component learns from existing data or from scratch by iteratively interacting with the network environment. HyMER approach is the first

that proposes a hybrid machine learning solution considering both energy efficiency and network performance in SDN.

As part of the end system aware module of the controller, we jointly address the energy efficiency of servers and network components. We propose a physical server utility-based metric named Ratio for Energy Saving of Physical Machines (RESPM) which measures how energy efficient the physical servers with respect to virtual machines residing in them.

Experiments are conducted on POX controller and Mininet network emulator using real topologies and traffic traces. Considering various metrics of interest and different types of SDN enabled switches, the results show that maximizing the RESDN value improves energy efficiency while maintaining acceptable network performance. In comparison to state-of-the-art utility-based heuristics, RESDN method performs up to 30% better ratio for energy saving, 14.7 watts per switch power saving, 38% link saving, 2 hops decrease in average path length and 5% improved traffic proportionality. The supervised component of HyMER achieves more than 65% feature size reduction and 70% accuracy in parameter prediction. The refine heuristics algorithm increases the accuracy of the prediction to 100% with 25X speedup as compared to the brute force method. Results show that HyMER achieves improvements in per switch power saving, link saving, and decrease in average path length.

ÖZETÇE

Yazılım Tanımlı Ağ (SDN) modeli, denetim ve yönlendirme düzlemlerininin ayrılması özelliğiyle programlanabilir ağ elemanları, ağ yönetiminde esneklik ve verimlilik avantajlarına sahiptir. Enerji maliyetleri, ağlardaki genel maliyetlere büyük oranda katkıda bulunurken, enerji verimliliği modern ağ mekanizmaları için önemli bir tasarım gereksinimi haline gelmiştir. Bununla birlikte, enerji verimli çözümler tasarlarken enerji verimliliği ve ağ performansı arasındaki dengenin dikkate alınması önemlidir. Bu tez çalışmasında, enerji verimliliği için trafik farkında, makine öğrenmeye dayalı ve son sistem farkında SDN denetleyici modülleri önerilmektedir.

Kontrol ünitesinin ilk modülü olarak, bağlantı kullanım aralıklarına göre enerji verimliliğini ölçen SDN Enerji Tasarrufu Oranı (RESDN) adlı yeni bir enerji verimliliği ölçütü önerilmektedir. Ağın RESDN değerini artırma amacıyla tamsayılı programlama formülasyonu ve yöntemi sunulmaktadır. RESDN yaklaşımı, bağlantıların harcadıkları enerji miktarı açısından karlı bir şekilde nasıl kullanıldığını ölçmesi yönüyle özgündür.

Kontrol ünitesinin ikinci modülü olarak, enerjiyi verimli kullanan yönlendirme için yeni bir hibrit makine öğrenme tabanlı HyMER sistemi önerilmektedir. HyMER denetimli ve pekiştirici öğrenme modellerinin avantajlarını birleştirmektedir. Denetimli öğrenme bileşeni, özellik çıkarma, eğitim ve testlerden oluşur. Pekiştirici öğrenme bileşeni mevcut verilerden veya sıfırdan ağ ortamıyla yinelemeli olarak etkileşerek öğrenir. HyMER yaklaşımı, yazılım tanımlı ağlarda enerji verimliliği ve ağ performansı için bir karma makine öğrenme çözümü özelliğiyle yenilikçidir.

Kontrol ünitesinin son sistem farkında modülünün bir parçası olarak, sunucuların ve ağ bileşenlerinin enerji verimliliği birlikte ele alınmaktadır. Fiziksel sunucuların, içerdikleri sanal makinelere göre enerjinin ne kadar verimli olduğunu ölçen, Fiziksel Makinelerin Enerji Tasarrufu Oranı (RESPM) adlı bir fiziksel sunucu fayda tabanlı ölçüt önerilmektedir.

Deneyler, gerçek topolojiler ve trafik izleri kullanılarak POX kontrol ünitesi ve Mininet ağ emülatöründe gerçekleştirilmiştir. Çeşitli başarım ölçütleri ve farklı SDN özellikli anahtar türleri dikkate alınarak elde edilen sonuçlar RESDN değerini en üst düzeye çıkarmanın kabul edilebilir ağ başarımını sağlarken enerji verimliliğini artırdığını göstermektedir. En son teknolojiye dayalı fayda tabanlı sezgisel araştırmalara kıyasla RESDN yöntemi, enerji tasarrufu için %30'a varan oranlarda başarım, anahtarlama gücü başına 14,7 watt, %38 bağlantı tasarrufu, ortalama yol uzunluğunda 2 atlama azalması ve trafik orantılılığında %5'lik iyileştirme sağlamıştır. HyMER denetimli öğrenme bileşeni, %65 oranında özellik boyutu azalması ve parametre tahmininde %70'den fazla doğruluk sağlamaktadır. Arıtma sezgisel algoritması, 25X hızlanma ile tahminin doğruluğunu %100'e yükseltmiştir. Sonuçlar, HyMER yönteminin anahtar başına güç tasarrufu, bağlantı tasarrufu ve ortalama yol uzunluğunda azalma sağladığını göstermektedir.

Acknowledgments

First and foremost, praised be the name of the Almighty God for giving me the motivation, strength and health to study. I want to extend my sincere gratitude to my thesis advisor Prof.Dr. Öznur Özkasap for her unreserved support, guidance, and encouragement from the beginning to the end of this thesis. I am mostly indebted to her for the research skills and professional ethics I acquired at Koç University and most importantly I want to thank her for being understanding to the problems I faced during my Ph.D. studies. I am grateful to my thesis progress committee members Prof.Dr. A. Murat Tekalp and Asst.Prof.Dr. Didem Unat who shared their invaluable experience, time and support in closely monitoring my thesis work. I am thankful to my thesis committee members Assoc.Prof.Dr. Berk Canberk and Assoc.Prof.Dr. Müge Sayıt for their invaluable time and support on evaluating my thesis.

Special thanks to my mother Genet Wudineh, and to my father Gizachew Assefa who motivated me to love mathematics and physics, who is my role model, whom I lost in the middle of my study, may his kind soul rest in peace. Your memory and advise will always remain with me. A big thanks to my in-laws Taddesse Delle, Dibabe Geletu, Bereket Tadesse, Mekrez Tadesse for having my back and taking care of my family when I was not around.

Ph.D. allowed meeting wonderful friends from all over the world. Thank you all for the good memories we have as students. Special thanks to my long-time friend Seyhan Uçar, who has been of great help from the day I started masters in Izmir. Thank you İpek Kızıl for your friendship which is even extended to my kids and wife besides

your collaboration in my research. I acknowledge the warm and sincere friendship I have with Karim Sonbol and Shahid Iqbal. I am blessed to have a concerned fellow, Dzenaida Gicic, Sanaz Taheri, Yahya Hassanzadeh, Umur Alp and Buket Yuksel. Faizan Ali and Waris Gill thank you especially for the constructive feedback on my presentations. Yalçın Şadi, Devriş İşler, Irfan and Kiran, Khuram and Nida, Emre Şamci, Emre Köse, Cemil Cengiz thank you for your hospitality and friendship. To the many people at Koç University whom I have not mentioned your name but still feel privileged to know you, thank you all!

I am privileged to be surrounded by many Ethiopian and Eritrean friends which we have grown into a family. I am ever grateful for the love you bestowed upon me, the happy and sad moments we have shared, and especially my brothers and sister in the church who have set an example of a sacrificial life. Pr. Adinew Endrias & Meron Abera, Gezahegn Endaresaw and Meaza Adamsega, Tigist Teshome & Emeka Auguchume, Betty & Jonathan, Rim & Meklit, Mekonen Aga &, Wudnesh , Henock Teka & Liya Matheos, Emmanuel Tamiru & Mihiret Girma, Tadiyos & Yodit, Abriham Salo, Samuel Mulugeta, Hundessa Dechassa, Asfaw Mekonen, Mimi, Wuda Tebikew, Simenesh Feleke, Peniel Tadesse, Haimanot Girma, Tina Tanga, Zaza T., Elita H, Fikir S, Dehnye & Gizaw family, Aster M., Hanna Z, Rahel Z, Azeb D., Sarno & Helen Sileshi, Yeabi & Getinet , Gitim & Samuel Kawiso, Lidiya Amanuel, Mihiret Genet, Naomi , Gashaw, and all brother and sisters at KGC-Istanbul.

I would like to thank Prof. Emine Yılmaz, my first advisor at Koç University, who accepted me as her Ph.D. student. I want to thank Koç University for offering me full scholarship to pursue my Ph.D. I also want to thank Emine Büyükdurmuş, Elif Tüysüz, and Gözde from GSSE, for making my stay at the university easier. I am grateful for Türkan Eryiğit from the Deans office and Ayşe abla for making feel at home in the Engineering lounge.

This thesis work was partially supported by the COST (European Cooperation in Science and Technology) framework, under Action IC0804 (Energy Efficiency in

Large Scale Distributed Systems), and by TUBITAK (The Scientific and Technical Research Council of Turkey) under Grant 109M761.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Software Defined Networking (SDN) paradigm has been attracting an increasing research interest, with its key concepts of control plane and data (forwarding) plane separation and logically centralized control by means of programmable network devices [1–4]. The idea behind SDN paradigm, depicted in Figure 1.1, is eliminating the tight coupling between control and forwarding functions in traditional network design, and hence the drawbacks of cumbersome network configuration and limited flexibility to changing requirements [5]. The management and control of the network is done in the control plane. The controller has the functionality of configuring the forwarding tables (in other words, flow tables) of switches. The set of connected switches constitutes the data plane of the SDN architecture. The sole responsibility of the data plane is to forward packets based on the forwarding rules installed in the flow tables of the switches by the controller.

SDN has been deployed in a diverse set of platforms ranging from institutional networks to data center networks. It promises several advantages such as flexibility without sacrificing forwarding performance, high efficiency through optimized routing, ease of implementation and administration, and cost reduction. The energy

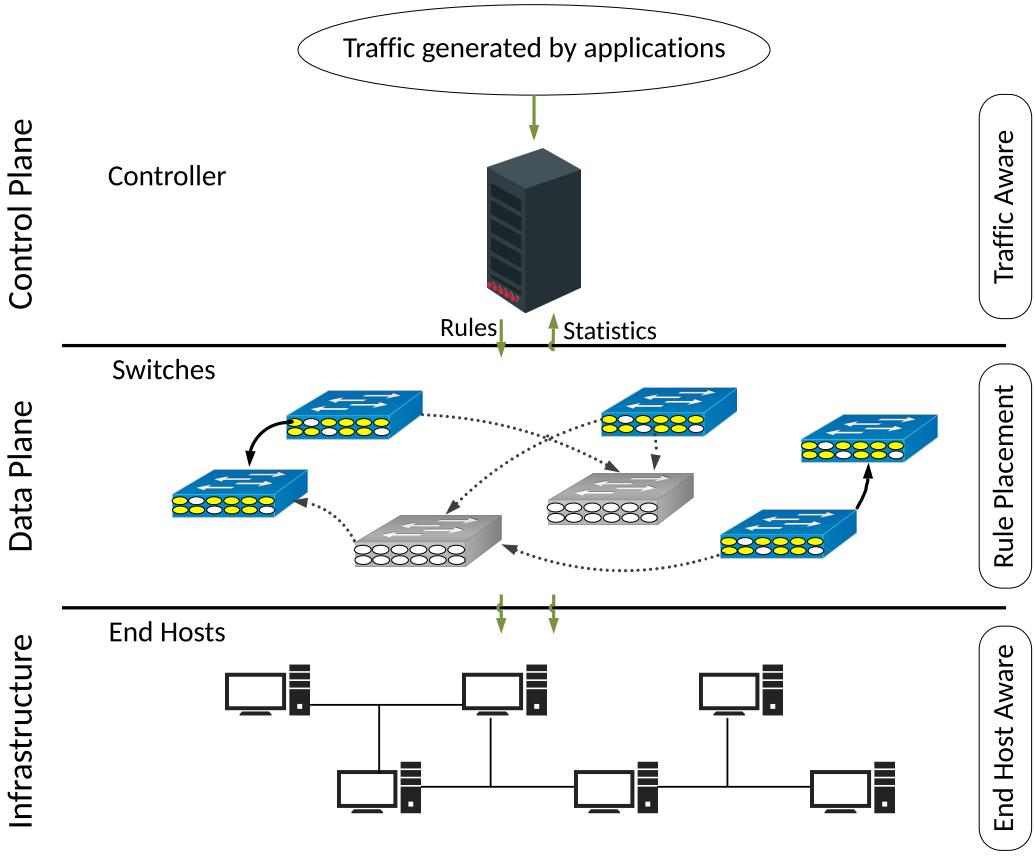


Figure 1.1: Software Defined Networking Architecture

consumption constitutes a significant portion of overall information and communication technology costs [6–8]. Energy constitutes more than 10% of OPEX (operating expenses) of an ICT service provider. SMARTer 2020 report predicts that the electricity cost of the cloud data centers will increase by 63% in 2020 [9]. Several survey studies have been conducted on reducing energy costs in different network settings such as P2P systems [10], Network Function Virtualization (NFV) [11], cloud data centers [12], and wireless sensor networks [13].

We address energy optimization that can be applied at various levels of the SDN architecture, or SDN itself that can be used as a means of energy saving. Energy saving in SDN can be addressed algorithmically or through hardware-based improvements. Software-based solutions are applied on the controller. The three energy

saving capabilities that can be addressed algorithmically are traffic aware, end system aware and rule placement. We provide an extensive survey and classification of recent developments in energy efficiency for SDN along with optimization models for each group of solutions [5, 8].

1.2 Approach and Scope of the Thesis

Energy efficiency in software defined networking can be addressed either by software-based methods or hardware improvements. In this thesis, the focus is on software-based solutions. There exist novel controller modules that we utilize to achieve energy efficiency, namely, traffic aware, machine learning based, and end system aware.

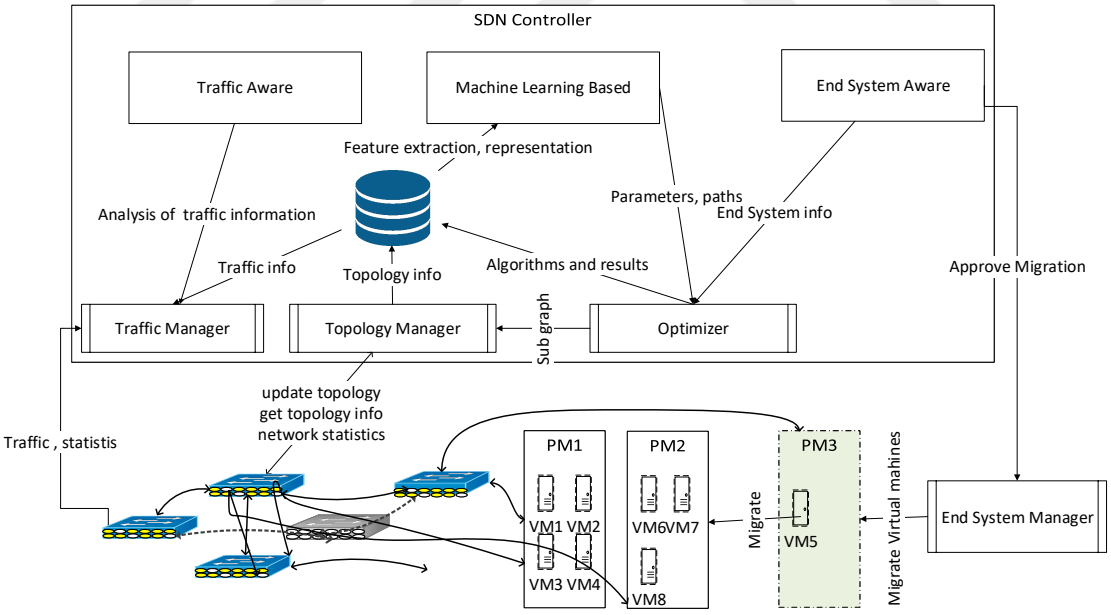


Figure 1.2: Overview of the proposed SDN controller modules

A high-level description of the approaches in the thesis is demonstrated in Figure 1.2. Our proposed approaches to energy saving in SDN are traffic aware, machine learning based, and end system aware solutions. The components of the contributions can be implemented as a module in any SDN controller. In this thesis, the modules

are designed and then implemented in a POX controller. The data plane and the infrastructure planes are emulated using Mininet [14]. The traffic manager, topology manager, and optimizer are responsible for managing traffic, controlling the network topology, and run heuristics respectively. The repository is a database that keeps track of every activity in the network. It stores traffic information, topology information, network statistics, and results of heuristics. The information stored in the repository is further used by the machine learning module of the controller.

The traffic aware module proposes traffic aware framework for energy efficient routing in SDN, an energy efficiency metric named Ratio for Energy Saving in SDN (RESDN) that quantifies energy efficiency based on link utility intervals. We provide integer programming formulations and method for maximizing the RESDN of the network.

As machine learning module of the controller, we propose HyMER: a novel hybrid machine learning framework for traffic aware energy efficient routing in SDN which has a supervised and reinforcement learning components. The supervised learning component consists of feature extraction, training, and testing. The reinforcement learning component learns from existing data or from scratch by iteratively interacting with the network environment.

The end system aware module of the controller, we jointly address the energy efficiency of servers, network components, and the performance of virtual machine migration. We propose a physical server utility-based metric called Ratio for Energy Saving of Physical Machines (RESPM) which measures how energy efficient the physical servers with respect to virtual machines residing in them. We also present an integer programming formulation to jointly maximize network energy efficiency and RESDN values.

1.3 Original Contributions

As an effort of understanding the energy efficiency problem in SDN, an investigation on energy saving capabilities in SDN has been performed [5, 8]. The contributions of our extensive survey work are as follows.

- We address the energy efficiency capabilities that can be utilized in SDN. Energy efficiency can be addressed algorithmically or by hardware design.
- We propose a taxonomy of software-based energy efficient solutions in SDN and general optimization models for each subcategory, and for each model we present the objective function, the parameters to be considered and constraints that need to be respected.
- We present key characteristics of state-of-the-art solutions for each category, their advantages, drawbacks, and provide comparisons with the general models.

The contributions of the traffic aware and proportional energy efficient routing module in software defined networking are as follows [15–18].

- We propose a traffic proportional energy efficient framework for SDN. The components in the framework deal with traffic monitoring, topology management, and optimization.
- We propose a novel energy efficiency metric RESDN: the ratio for energy saving in SDN that quantifies energy efficiency based on link utility intervals. To the best of our knowledge, the approach is unique as it measures how links are profitably utilized in terms of the amount of energy they consume with respect to their utility.
- We develop Integer Programming (IP) formulation with the objective of maximizing RESDN and propose heuristics algorithm MaxRESDN for achieving the objective.

- We conduct extensive experiments on Mininet network emulator and POX controller using real network traffic traces and present comparative quantitative analysis of MaxRESDN method. The performance metrics of interest are switch power consumption, RESDN value, percentage of links saved, average path length, throughput, delay, and traffic proportionality.

- We simulate the power consumption of two hardware switches (NEC and Zodiac FX) and one virtual switch (Open vSwitch-OvS) that are OpenFlow enabled, and conduct comprehensive experiments to measure the power consumption of our proposed MaxRESDN heuristic algorithm.

The contributions of the machine learning module of the controller are as follows [19, 20].

- We propose a hybrid three module machine learning framework, namely HyMER, for traffic proportional energy saving in SDN. The modules are Traffic Manager, Topology Manager, and Learning Machine. To the best of our knowledge, this is the first work to consider the learning machine as a module in an SDN controller for energy saving and network performance.

- Most of the machine learning approaches proposed for SDN are for traffic classification, routing, intrusion detection, or attack prediction. To the best of our knowledge, our HyMER framework is the first in applying machine learning to energy saving and network performance combined using both supervised and reinforcement learning.

- We propose a full-fledged supervised machine learning method that starts from feature extraction, applies feature reduction, and performs testing. Our results indicate more than 65% feature size reduction using Principal Component Analysis (PCA). The supervised component predicts the link utility interval parameters with an accuracy of more than 70%. The proposed refine heuristics

converges the predicted values to the optimal values with a speedup of 15X to 25X as compared to the brute force approach.

- We also propose a reinforcement learning method that minimizes energy consumption while keeping acceptable performance for dynamic routing in SDN. To the best of our knowledge, HyMER reinforcement component is the first to model both network performance and energy efficiency simultaneously. The reinforcement method converges to the maximum energy saving with a minimum of 100 to a maximum of 270 episodes. Episodes are the number of iterations which is the measure or time it takes for the reinforcement learning agent to reach the terminating state.

- We also demonstrate that combining the supervised and reinforcement methods not only does capture the dynamic change more efficiently but also increases the convergence speed. To the best of our knowledge, in the context of SDN, our approach is the first to combine supervised and reinforcement learning to jointly achieve energy saving and network performance. Initializing the reinforcement learning with the outputs of the supervised component speeds up the convergence by 2X on average.

- Experiments are conducted with Mininet and POX controller using real word network topologies and traffic traces from SNDLib. In particular, Abiline, GEANT, and Nobel-Germany topologies and dynamic traffic traces are utilized. Switch power consumption is simulated using the SDN enabled switch NEC [21].

The contributions of the end system aware module of the SDN controller are as follows.

- We propose an energy efficiency metric RESPM which is based on physical machines utility interval. We also extended our network component energy

efficiency and performance metric RESDN to be used jointly with RESPM. Unlike other approaches which focus on server energy efficiency or network component efficiency, our approach consider both simultaneously.

- We presented an IP formulation with objective to jointly maximize the RESPM and RESDN.

- Experiments are conducted on real world data center virtual machine placement traces.

1.4 Organization

The remainder of the thesis is organized as follows. Chapter 2 presents an overview of energy saving capabilities in SDN. Chapter 3 presents traffic-aware solutions, a general optimization model and methods in the literature. The end system aware solutions, multiple objective optimization model and techniques are presented in Chapter 4. Chapter 5 presents the proposed traffic-aware framework, IP formulation, and link utility-based heuristic algorithms, that are not only general in their applicability but also balances the trade-off between energy saving and performance.

Chapter 6 introduces a novel energy utility interval-based metric Ratio for Energy Saving in SDN (RESDN), IP formulation to maximize the RESDN value of a network, and MaxRESDN heuristics. A comprehensive experimental comparison of MaxRESDN with similar utility-based heuristics and analysis of the utility interval parameters is presented in Chapter 7.

Chapter 8 describes our proposed hybrid machine learning framework for energy efficient routing in SDN (HyMER). Chapter 9 presents comprehensive experimental analysis of HyMER in comparison with similar approaches.

A physical machine utility interval metric called Ratio for Energy Saving for Physical Machines (RESPM), an IP formulation that jointly minimizes the energy saving of physical machines and network components along with experiments on real world

data center traces are presented in Chapter 10. Chapter 11 concludes the thesis and provides future directions.

Chapter 2

Overview of Energy Saving in Software Defined Networks

2.1 Energy Saving Capabilities in SDN

In software defined networking, energy optimization that can be applied at various levels of the network architecture, or SDN itself that can be used as a means of energy saving. Figure 2.1 shows the categories of energy saving approaches in SDN. Energy saving in SDN can be addressed algorithmically or through hardware-based improvements. Software-based solutions are applied on the controller. The three energy saving capabilities that can be addressed algorithmically are traffic aware, end system aware and rule placement.

Traffic awareness is the capability to make the energy consumption proportional to the traffic volume. End system awareness on the other hand uses SDN to save energy by capitalizing on virtual machine placements and migrations. Most energy aware routing algorithms do not assume the limited rule space inside switches. Rule placement techniques address the energy saving from the meaning of the rules, and compressing them in way they save space.

Traffic aware energy efficiency approaches are inspired by the fact that network components are often under utilized. The key principle is to turn on or turn off

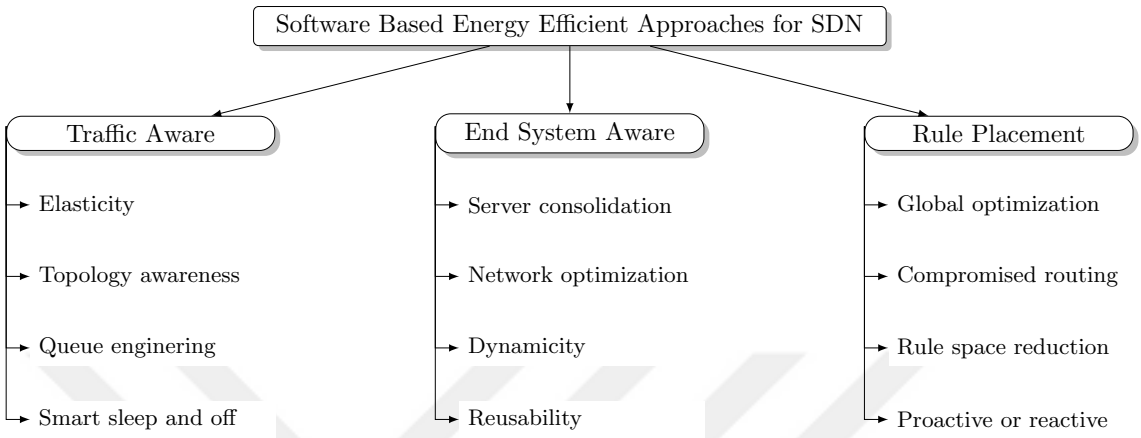


Figure 2.1: Classification: Software-Based Energy Efficiency Approaches in SDN

network components (i.e., SDN forwarding switches) based on the traffic load. For instance, when the traffic load is low (e.g., during night times) this approach has the potential to save up to 50% of the total energy consumption [22]. Typically, an elastic structure is used to represent the network components that can grow and shrink with the dynamic traffic load. The key challenge is to determine which components to turn on and which components to turn off without compromising the required quality of service (QoS) [22–34].

As presented in Figure 2.1, desirable properties of a **traffic aware** controller are elasticity, topology awareness, queue engineering, and smart sleep on and off. **Elasticity** is the ability to dynamically enlarge or shrink the number of network components used in response to traffic. **Topology awareness** provides an extra benefit of using formulations and solvers that can be tailored to any specific topology. The hierarchically organized fat-tree is the widely used topology in data centers. A prior knowledge to how the components are organized and their capacity allows us to use alternative routes by avoiding energy critical paths. **Queue Engineering** techniques applied to each port for the packet arrivals give additional port level traffic monitoring capability. **Smart sleep and off** is the ability to turn on/off ports of switches, links, or the entire switch to save energy in response to traffic.

End system aware energy saving solutions use the practice of turning off underutilized physical servers and running their tasks on a fewer number of servers in SDN based data centers [35–42]. Specifically in data centers, the SDN model is used to form an overlay connecting virtual machines. The overlay is a list of SDN enabled switches and links. The actual connection is of course with the physical machines which host the virtual machines.

Desirable properties of an end system aware solution are depicted in Figure 2.1. **Server consolidation** is the technique of minimizing the number of physical machines that are active by placing as many virtual machines as possible to fewer physical machines. **Network optimization** on the other hand minimizes the transport cost of migrating virtual machines from one physical machine to the other one. **Dynamicity** is the ability to apply both server consolidation and network optimization online instead of offline. The need for online migration of virtual machines and network optimization in response to traffic and workload is of great value. **Reusability** is the characteristic of re-booking resources after they are released from prior works. This comes from the fact that some resources can be used for longer period of time as compared to others.

Rule placement techniques focus on how to place the rules in the forwarding switches. Given the network policies and end-point policies, the controller provides a mechanism to convert the high level policies into switch understandable rules. Rule placement is an NP-hard problem and hence requires heuristic based solutions. Although heuristic based approaches do not guarantee optimal solution, they typically offer close to optimal results. And their efficiency depends on the complexity of the constraints [43–48]. Some of the constraints considered for this problem are the maximum number of rules a switch can hold, the routing policy, and the topology. Under such constraints, rule placement approaches attempt to optimize routing.

Global view is the ability to use entire network information, routing policy, and end-point policy to decide which rules to place on which switch. **Compromised routing** is the ability to find alternative power efficient routes at the cost of per-

formance. However, the trade off between compromised routing performance due to extra longer paths and its impact on QoS should be taken with great care.The set of all rules that need to be installed on the switches constitute the rule space. **Rule space reduction** is the ability to reduce the rule space, hence, minimize the number of active links and or switches. **Proactive / reactive** is the ability to do energy efficient rule placement proactively before the packets arrive to the network, and respond reactively to newly arriving packets.

As far as the software-based solutions are concerned, the controller should be acquainted with the necessary statistical information about the status of the network devices and end systems attached, in order to devise a way for energy efficient networking. A full fledged energy efficient controller needs to consolidate end systems, network device optimization, and also be able to reduce the rule space.

2.2 Background and Terminology

In traditional networking, the data plane and the control plane reside in the switch. The control plane populates the rules required for forwarding into the forwarding table. Whereas, the forwarding plane reads the forwarding table to forward packets. The control decisions taken at a given switch determines the next hop in the network that each packet needs to be forwarded. The intelligence in networking which includes security, forwarding, and load balancing at switch level. Since the control plane and the data plane are highly coupled in every networking component, adapting any change in the networks is cumbersome because it demands every component of the network to be configured.

SDN, on the other hand, is a novel paradigm in networking with the key design principle of the control plane and the data plane separation. This design principle, where the data plane resides at the switches and the control plane resides in the controller, provides flexibility network management. The forwarding table is inside the networking components, but the rules are created and pushed to each switch by

the controller. The prominent advantage of SDN is that it provides a global view of the network to be controlled and monitored centrally. The switches in SDN only do packet forwarding. The control is done by the logically centralized controller. SDN makes the network programmable, and enables changes done on the controller to propagate to the entire network.

Table 2.1: Parameters and descriptions

Symbol	Description
$\mathbb{Z}$	Set of switches in the network where $Z_i \in \mathbb{Z}$ represents switch i
$\mathbb{E}$	Edge or links where $e_{ij} \in \mathbb{E}$ represents the link between switches Z_i and Z_j
W_{ij}	Bandwidth of the link connecting switches Z_i and Z_j
S_i	$=\begin{cases}1, & \text{if switch } Z_i \text{ is active,}\\ 0, & \text{otherwise.}\end{cases}$
CS_i	Power consumption of switch Z_i
C_{ij}	Power consumption of the link connecting switches Z_i and Z_j
$\mathbb{F}$	Set of flows where $f \in \mathbb{F} = (sr, ds, \lambda_f)$
$f = (sr, ds, \lambda_f)$	A flow f with source , destination and packet rate
F_{ij}	$=\begin{cases}1, & \text{if flow } f \text{ passes through edge } e_{ij},\\ 0, & \text{otherwise.}\end{cases}$
$\mathbb{P}$	Set of physical machines
$\mathbb{V}$	Set of virtual machines
$\mathbb{R}$	Vector of resource types
$\mathbb{R}p$	Set of resources available on physical machines represented as a matrix
$\mathbb{R}v$	Set of resource demands of virtual machines represented as a matrix
X_{ij}	$=\begin{cases}1, & \text{if virtual machine } j \text{ is placed on physical machine } i,\\ 0, & \text{otherwise.}\end{cases}$
PM_i	$=\begin{cases}1, & \text{if physical machine } i \text{ is on,}\\ 0, & \text{otherwise.}\end{cases}$
Pr_i	Amount of resource type r available on physical machine i
V_j^r	Amount of resource type r required by virtual machine j
$\mathbb{Q}$	Traffic matrix between virtual machines
q_{ij}	Amount of traffic between virtual machines i and j
b_{ij}	Number of switches that the traffic between physical machines i and j traverses
H_{in}	Set of ingress hosts
H_{eg}	Set of egress hosts
$\mathbb{A}$	Allocation matrix, $a_{if} = \begin{cases}1, & \text{if rule representing flow } f \text{ is installed on switch } Z_i,\\ 0, & \text{otherwise.}\end{cases}$
L_{ij}	$=\begin{cases}1, & \text{if edge } e_{ij} \text{ active,}\\ 0, & \text{otherwise.}\end{cases}$
G_i	The maximum number of rules switch Z_i can store

Table 2.1 presents notation used in this study for formulating the energy efficiency problem. The symbols are listed in the order they are used in the study. The set of

switches are denoted by $\mathbb{Z}$ where each switch (Z_i) is a forwarding device with the role of forwarding packets based on their flow information. S_i is a binary variable with value 1 if the switch Z_i is turned on, or 0 otherwise. The total power consumption of Z_i is denoted by CS_i.

Switches are connected to one another by links.

A **link** is active if it is transmitting packets between the two ports of the switches it connects. L_{ij} is a binary variable with value 1 if the link connecting switches i and j is active, or 0 otherwise. The power consumption of a link connecting switches i and j is denoted by C_{ij}. G_i and W_{ij} are the maximum number of rules that can be installed in switch i and the bandwidth of the link connecting switches i and j, respectively.

A network in SDN is composed of forwarding switches connected by links, which is transmission media for data among end systems. An **end system** is a physical or a virtual machine where services and applications are running on. The set of physical machines and virtual machines are denoted by $\mathbb{P}$ and $\mathbb{V}$, respectively. Each physical machine has a limited set of resources $\mathbb{R}p$. A virtual machine i has set of hardware or software demand $\mathbb{R}v$ to be able to run on a physical machine. X_{ij} is a binary variable with value 1 if virtual machine j is placed on physical machine i. PM_i is a binary variable with value 1 if physical machine i is on or 0 otherwise. $\mathbb{R}p$ and $\mathbb{R}v$ are represented as matrix of resources available on physical machines and resource demands of virtual machines, respectively. Vector $\mathbb{R}$ represents the resource types. The matrix $\mathbb{R}p$ and $\mathbb{R}v$ have the dimensions of $|\mathbb{P}|$ by $|\mathbb{R}|$ and $|\mathbb{V}|$ by $|\mathbb{R}|$, respectively. A set of virtual machines is placed on a physical machine if and only if the sum of the resource demands of virtual machines can be met by the available resources on the physical machine. The traffic matrix between virtual machines is denoted by $\mathbb{Q}$, where q_{ij} represents the traffic between virtual machines i and j measured in bps. b_{ij} is the number of switches the traffic between physical machines i and j traverses.

A switch contains a **flow table** or set of **flow tables**. The fields of a flow table may vary from vendor to vendor. OpenFlow 1.0 switch specification presents 12

packet matching fields (header fields), counter, and action. The matching fields have information that can be compared with packets arriving to the switch. Per-flow, per-switch, and per-port counters are maintained in the flow table. The action field is an instruction of what to be done with the corresponding packet matching the header field. According to OpenFlow 1.4 standard, a switch can have multiple tables and matching is done through pipeline process among the tables.

Packets flow through the network starting from their source towards destination. An important issue while managing the network is the level of granularity of the control. Packet level control makes the controller congested since a new rule should be created and pushed for every packet, whereas prefix-based matching reduces the control capability since several packets matching a prefix would be treated the same way.

As an intermediate level of granularity, flows are introduced [49].

The set of flows is denoted by $\mathbb{F}$ where a flow is defined as a set of packets with the same source and destination addresses where the packets pass through the same route to reach the destination. H_{in} and H_{eg} are the set of ingress hosts and egress hosts, respectively. Ingress hosts are hosts where flows start from and egress hosts are hosts which are destinations of flows. λ_f denotes the packet rate of flow f. G_i is the maximum number of rules that can be installed in switch Z_i, respectively.

Rule in SDN is a predicate that describes a flow. An example of a rule predicate is destination IP=198.127.***.***. If a packet that has the destination IP address that starts with 198.127, its corresponding action will be taken. The controller populates the flow tables by pushing the rules to each switch.

The rules are stored with priority which is the matching precedence of the flow entry. A rule which all entries are wildcards is given the least priority value. For each packet arriving a switch, its flow information is extracted from the packet header and then looked up in the flow table. Then, the highest priority entry matching the packet is selected. $\mathbb{A}$ is an allocation matrix between flows and switches where a_{if} is a binary variable with value 1 if rule representing flow f is installed on switch Z_i or

0 otherwise.

An **action** defines the task to be performed for the packet that matches a rule. An action for a packet can be forward to the indicated port, drop the packet, default port to forward to the controller, modification of the packet header or any other action depending on the type of protocol used. The action associated with the first rule that matches the packet will be executed. If a packet does not match any rule in the flow table, then it is forwarded to the controller.

After a set of rules is installed for a packet, if another packet comes with the same header description, there is no need to push new rules, rather it uses the existing rules.

Chapter 3

Traffic Aware Energy Saving Optimization Model and Methods in SDN

3.1 Overview

Traffic in networking is defined as the amount of packets transmitted through a network at a given point in time. Traffic management involves dynamically analyzing, predicting and regulating the behaviour of the network devices to optimize the performance and QoS requirements. Examples based on usage of traffic are load balancing, performance optimization, security, access control, bandwidth management, and energy usage [50] [51].

The logically centralized controller in SDN has a global view of the network. The controller periodically gathers statistical information about the traffic, the status of the switches, the status of the links, the status of the end systems, and the topology of the network. Programmability of a network, a powerful property of SDN, enables the controller adapt to the changing environment. A change in the network environment can be caused by addition of a new host, failure of a network component, or modification of network policy. Network components work with full capacity in

all times regardless of the traffic demand. The power consumption of the network is not proportional to the volume of traffic [22]. With traffic aware energy efficiency approaches, energy consumption can be reduced by turning off some forwarding switches during low traffic load, or putting CPUs or ports at sleep mode. The solutions in this group have the potential to significantly improve energy efficiency in SDN. For data centers, traffic aware approach achieves power savings of up to 50% during low load periods [22].

3.2 Traffic Aware Model

Based on the review of various models used to capture traffic proportional energy consumption, we propose a general optimization model that identifies energy capabilities in SDN from traffic point of view, where the major energy saving components are considered as the links and the switches. The model jointly minimizes the number of switches and the number of links used to accommodate the given traffic. It is tailored to include additional parameters and constraints. For simplicity, the parameters of the optimization model refer to snapshot of the network state. Given the flows which represent the traffic, the problem is defined as allocating links and switches for each flow while minimizing the total number of active links and switches. Inspired by the approaches of [22,31,33,52], we use multi-commodity flow problem formulation. Each flow is treated as a commodity with source, destination, and flow rate. Our model is inspired by the formulation of [22] and the objective of the model is to minimize the power consumption of the links and switches used to handle the flows. The other approaches in this category consider additional parameters like time, bandwidth awareness, and redundant path elimination.

In traffic aware model, the network is represented as an undirected weighted graph $\mathbb{G}= (\mathbb{Z},\mathbb{E})$ where $\mathbb{Z}$ is the set of switches and $Z_i \in \mathbb{Z}$ represents switch i and $e_{ij} \in \mathbb{E}$ represents that there is link between switches Z_i and Z_j. The weight W_{ij} corresponds to the bandwidth of the link connecting switches Z_i and Z_j. Let binary variable S_i

denote the status of switch Z_i such that

$$S_i = \begin{cases} 1, & \text{if switch } Z_i \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

CS_i and C_{ij} are power consumption of switch Z_i and the link e_{ij} measured in watt.

Traffic in the network is represented by set of flows $\mathbb{F}$ where $f \in \mathbb{F}$ is defined as $f{=}(sr, ds, \lambda_f)$. sr and $ds \in \mathbb{Z}$ are the source and destination switches and λ_f is the rate of flow f measured in bytes per second.

$$F_{ij} = \begin{cases} 1, & \text{if flow } \ f \text{ passes through edge } e_{ij} \\ 0, & \text{otherwise} \end{cases}$$

The multi-objective function (equation 3.1) minimizes the sum of the energy consumption of the switches and the links. The first item in the objective function, the sum of $F_{ij} * C_{ij}$, refers to the total energy consumption incurred by all flows using edge e_{ij}. The second item is the total power consumption of all active switches in the network. The objective function jointly minimizes the sums subject to the constraints described next.

$$\text{minimize } (\sum_{\forall f} \sum_{\forall e_{ij}} F_{ij} * C_{ij} + \sum_{\forall S_i} S_i * CS_i) \tag{3.1}$$

$$\text{subject to } \sum_{\forall f} F_{ij} * \lambda_f \leq W_{ij} \ , \forall e_{ij} \tag{3.2}$$

$$\sum_{\forall f} F_{ai} = \sum_{\forall f} F_{ib} \ , \ Z_i \neq sr, ds \in f_{sr,ds,\lambda_f} \tag{3.3}$$

$$F_{mj} = F_{in} \ , \ Z_m = sr \ , \ Z_n = ds \ , \ \forall e_{mj} \ , \ \exists e_{in} \tag{3.4}$$

$$F_{ij} \leq S_j \ , \ \forall Z_j \in \mathbb{Z} \tag{3.5}$$

$$F_{ij} \leq S_i \ , \ \forall Z_i \in \mathbb{Z} \tag{3.6}$$

$$S_i \leq \sum_{\forall f} [F_{ij} + F_{ji}] \ , \ \forall Z_i \in \mathbb{Z} \tag{3.7}$$

The constraint given in Equation 3.2 states that the total rate of flows between two switches should not exceed the link capacity. Constraint specified in equation 3.3 states that the number of flows entering and leaving for switches which are neither destination nor sources of a flow should be equal. Constraint on equation 3.4 assures a flow entering from source switch should reach the destination switch. The constraints on equations 3.5, 3.6, and 3.7 maintain the correlation between switches and links using the switch state variable and flow-link variables. While constraints 3.5 and 3.6 state that no flow should use a link connected to an inactive switch, constraint 3.7 states that if no flow is passing through the links connected to a given switch, then the switch is switched off. Given the formulation and the parameters of the model, the output of the optimizer is list of links and active switches for each flow $f \in \mathbb{F}$.

3.3 Methods

Table 3.1 shows the summary of traffic aware solutions for energy efficient SDN. The experiment column presents how the proposed solution is tested. The testing environments are testbed, network simulation tools (ns-2), and emulation software (mininet, OMNeT++). The traffic traces used for the test are real world traces (R), artificially generated (A), or both (R&A). The topologies used in the evaluations are fat-tree, BCubic, CERNET, German, SNDLib provided traces. Controller column indicates the name of the controller used in the experiment (NOX, POX, FL-Floodlight, RY-RYU, and OD-OpenDaylight). The objective of the optimization is to minimize the number of active switches, links/ports or both. QueueEng column presents if a given approach uses switch level queue engineering techniques. The symbol '-' means unknown for Traffic, Topology, and Controller columns, and means does not apply in case of Switch, Link/port and QueueEng columns.

As shown in Table 3.1, most of the traffic aware energy efficient methods focus on minimizing the number of active links primarily and switches secondly or both. Testing energy efficient methods in real time on an actual network equipment is not timely

and economically feasible. That is the main reason why researchers resort to create small-scale testbeds or use simulation/emulations environments. Queue engineering is the least used technique and it mostly complements the active link/switch minimizing methods. The network topology and traces provided on SNDLib are widely used data sets in measuring energy efficiency in SDN.

Table 3.1: Traffic Aware Techniques for Energy Efficiency

Approach	Experiment				Objective		QueueEng
	Environment	Traffic	Topology	Controller	Switch	Link/ Port	—
ElasticTree [22]	Testbed	R	Fat-tree	NOX	-	✓	-
Carrier Grade [23]	Testbed	R	German	NOX	-	✓	-
REsPoNse [24]	NS-2	R	-	-	✓	-	-
CARPO [25]	Testbed	R	Fat-tree	-	-	✓	-
EnableOpenflow [26]	Mininet	R& A	Fat-tree	NOX	-	✓	-
RA-TAH [27]	Testbed	R & A	Fat-tree	POX	-	✓	✓
Re-Routing [28]	Testbed	R & A	CERNET	-	-	-	✓
Dynamic TA [53]	Testbed	R & A	Mesh	-	✓	✓	-
TE based [29]	Testbed	R & A	CERNET	-	✓	✓	-
NetFPGA QE [30]	Testbed	-	-	-	✓	-	✓
GreenRE [31]	Testbed	R& A	SNDLib	-	✓	✓	-
Bandwidth-aware [52]	OMNeT++	R	Bcubic & Fat-tree	-	-	-	✓
GreenSDN [32]	Mininet	R	Bcubic & Fat-tree	POX	-	✓	✓
Flow Schedule [54]	NS-3	A	Fat-tree	-	✓	✓	-
OpenNaas [33]	Testbed	-	-	RY, FL, OD	✓	✓	✓
Orchestrate [34]	Mininet	R& A	-	POX	-	✓	✓
Resource-aware [55]	Testbed	A	Fat-tree	-	-	✓	-
Multiple Contoller [56, 57]	Testbed	R	SNDLib	-	-	✓	-
5G [58]	Testbed	R	SNDLib	-	-	✓	-
Utility-based [17]	Mininet	R	GEANT	POX	-	✓	-
FLOWP [59]	OPNET	A	Fat-tree	-	✓	-	-
TA framework [16]	Mininet	R	SNDLib	POX	✓	✓	-
SDN /Ethernet [60]	Testbed	R	SNDLib	-	✓	✓	-
MTSDPFPR [61]	Mininet	R	GEANT	Floodlight	-	✓	-

ElasticTree is a power management solution for data center networks which is implemented on a testbed consisting of OpenFlow switches [22]. The idea is to turn off

links and switches based on the amount of traffic load. Therefore, energy consumption of the network is made proportional to the dynamically changing traffic. It consists of three optimizers: formal model, greedy bin-packing, and a topology-aware heuristic. Each optimizer takes network topology (a graph), routing constraints, power model (flat), and traffic matrix as input, and outputs subset of links and flow routes.

The formal model formulates the power saving problem by specifying objective function and constraints. The objective function minimizes the sum of the total number of switches turned on and the number of links. The advantage of the formal model is that it guarantees a solution within some configurable optimum; however, the model only scales up to 1000 hosts.

The greedy bin-packing optimizer evaluates possible paths and chooses in left to right order manner, that is, the leftmost path is chosen first then the selection proceeds till the rightmost path. The optimizer improves the scalability of the formal model. This approach suffers the same problem as any of the heuristic techniques. However, solutions can be computed incrementally and can support on-line usage.

The topology-aware heuristic optimizer, on the other hand, splits the flow and finds the link subset easily. It is computationally efficient, since it takes advantage of a fat tree structure and takes only port counters to compute link subset. This approach uses Integer Programming (IP) to formalize the optimization problem. The drawback is degradation of performance because of turning on and turning off components. The model used in ElasticTree contains more constraints, including fat tree topology constraints, than the general model. Experimental results show that ElasticTree achieves energy savings of up to 50%. One drawback is that it does not consider the correlation among the flows.

In the technique of Carrier Grade [23], the focus is the energy efficiency and resilience characteristics of carrier grade networks. Related to energy efficiency, it is demonstrated that OpenFlow can reduce network wide network energy consumption and improve scalability. MLTE is implemented together with energy saving mechanisms such as controlled adaptive line rates at the switches [62]. For the resilience, it

is shown that OpenFlow can handle failures at the switches and the controller, and perform recovery with flow restoration. Similar to ElasticTree, energy savings of up to 50% are achieved.

REsPoNse is a framework that allows network operators to automatically identify energy-critical paths [24]. It investigates the possibility to pre-compute a few energy-critical paths that, when used in an energy-aware fashion, can continuously produce close-to-optimal energy savings over long periods of time. REsPoNse identifies energy-critical paths by analyzing the traffic matrices, installs them into a small number of routing tables (called always-on, on-demand, and fail-over), and uses a simple, scalable online traffic engineering mechanism to deactivate and activate network elements on demand. The network operators can use REsPoNse to overcome power delivery limits by provisioning power and cooling of their network equipment for the typical, low to medium level of traffic. An additional formulation added to our general model is identifying the energy critical paths and modeling the links power consumption separately and achieves an energy saving of 40%.

CoRrelation-aware Power Optimization (CARPO) algorithm dynamically consolidates traffic flows onto a small set of links and switches in a data center network, and aims at switching off idle network components to reduce energy consumption [25]. It consolidates traffic flows based on correlation analysis among flows. Another important feature of CARPO is to integrate correlation-aware traffic consolidation with link rate adaptation for maximized energy savings. The integration is formulated as an optimal flow assignment problem. A near-optimal solution is first computed using integer programming to determine consolidation and the data rate of each link in the data center network. A heuristic algorithm is used to find a consolidation and rate configuration solution with acceptable runtime overheads. The heuristic reduces the computation complexity. In addition to our general model, CARPO introduces parameters to represent correlation among flows. This effort extends the ElasticTree work to be flow similarity aware also achieves 50% efficiency.

A test platform for measuring and analyzing the energy consumption of Open-

Flow based data centers is presented in [26]. The approach used is both traffic aware and energy aware. The three energy saving capabilities covered are switch, link and port. Considering traffic information, the number of active links and switches is determined. The Adaptive Link Rate (ALR) technique and a routing algorithm similar to ElasticTree are used to determine a route for each flow. The power model of NetFPGA-based OpenFlow switch and used NOX controller are the building blocks of the test platform. Experimental results show that this approach saves 35% energy.

ALR based techniques aim at reducing the energy consumption of a link by scaling the rate of the link proportional to the link utility. Rate Adaptive Topology-Aware Heuristic (RA-TAH) solution utilizes both smart sleeping and power scaling of links to improve energy efficiency of data center networks [27]. A comprehensive survey on ALR techniques can be found in [63]. The work is an extension of [22]. This combined mechanism was deployed in a data center using Fat-Tree topology. The bounds on energy savings in low and high traffic utilization cases were analyzed. Analytical results show that the combined algorithm reduces energy consumption remarkably as compared to the conventional methods in case of high traffic with up to 48% energy saving. Unlike the general model we proposed, this technique only focuses on link utilization rather the switches.

A global power management algorithm based on routing traffic via alternative paths to balance the load of links is presented in [28]. A binary integer programming model is used for formulating the energy consumption of integrated chassis and line-cards. The chassis and the line-cards are put to sleep based on link utilization and delay of packets. Less utility of links and large amount of delay of packets give information that either the chassis or the line care or both can be put to sleep. A greedy heuristic algorithm is proposed. Experimental results with scenarios of synthetic topology and real life topology CERNET demonstrate the reduction of power consumption by up to 67%.

Motivated by the fact that energy consumption of routers remains constant even if the traffic volume changes from time to time, a dynamic algorithm is proposed

in [53]. The power consumption minimization problem is formulated using IP model. The objective function jointly minimizes the sum of the power consumption of the links and the switches. Four greedy heuristics algorithms are proposed to solve the problem, namely, shortest path first, longest path first, smallest demand first, and highest demand first. The algorithms are tested over two sample topologies, a campus network and mesh network for low, medium and high traffic. Experimental results show that the longest path first gives close to optimal energy saving of up to 35% and is better than the other three algorithms for both topologies.

Traffic aware energy efficiency techniques for cloud computing infrastructure using OpenFlow are presented in [29], where Data Manager (DM) is introduced to detect input traffic and control the states of switches. The proposed power manager dynamically controls and updates the operating modes of switches. Based on the information from the DM, the Clock Controller (CC) module changes the frequency of the switch to 0 MHz if the traffic is low. The experimental results show that the approach guarantees QoS. This approach is immune to packet dropping as compared to previous works and results show whole switch saving of about 30-35% on average.

The idea in [29] is tested on the NetFPGA platform. Such an approach extends the OpenFlow protocol to include additional information about the frequency each switch is operating. Using this, the controller would be able to allow switches to work at different frequencies. Experimental results show that the proposed switch saves up to 95% when running at low frequency mode. For a large frequency between 2611mW and 11576mW, it saves 22.5% energy in total [30].

GreenRE [31] uses redundancy removal (RE) to achieve energy efficiency where the motivation is stated as follows. Networks exhibit several redundant links while users access similar contents. Even though redundancy increases reliability, it also degrades the performance of the network. Instead of sending the same data through many different paths repeatedly, sending it through a single link increases the throughput and in effect reduces the load in the links.The links with no loads are subject to turn off. GreenRE presents a work which capitalizes on RE to reduce energy. Experiments

conducted on Orange Labs platform demonstrate that RE approach results in 30% less energy consumption. A very important problem addressed in this work is the number of RE enabled routers needed to preserve QoS and reduce energy at the same time. In contrast to the general model we presented, the objective function minimizes only the number of RE-capable routers while minimizing the links.

A bandwidth aware energy efficient technique for data center is presented on the work of [52]. The controller schedules traffic flows. Traffics can be divided into three statuses, active, queued, and suspended. There is no need for clock synchronization among servers since a centralized controller is responsible for scheduling. A time aware power consumption model of a switch is also presented. A linear programming model is used to capture energy formulation of the entire network. The objective function minimizes the number of active switches and ports which increasing the occupation ratio of ports per switch. Experimental results show that the bandwidth aware technique demonstrates 8.85% higher energy efficiency and lower completion time in Fat-Tree and BCube topologies as relative to the fair share routing.

Energy efficient approaches attempt to minimize the number of power consuming components in the network. The drawback is that QoS may degrade. At the same time there is a need to keep idle network components for fault tolerance and restoration. Restoration and energy efficiency aim at achieving opposite objectives. Restorable Energy Aware Routing with Backup Sharing (REAR-BS) combines both energy efficiency and restoration into a single problem [64]. A non-linear formulation is used to minimize the number of active links under constraints of maximum utilization and restoration. However, the non-linear formulation is converted to its linear form to get a minimum bound to the optimal solution. An algorithm, Green Restorable Algorithm (GreRA) is proposed to solve the REAR-BS NP-hard problem. Simulation results show that GreRA reduces the energy usage by 15% to 50%.

The actual experimentation of energy efficient techniques is costly since it demands operational network. GreenSDN brings an energy efficiency simulation capability on the well known and widely used network emulator Mininet [14] and POX controller

[32]. The simulation tool is tested for both traditional techniques like ALR, switch chip level, switch level, and network-wide scope. The contribution of the study is that it is the first energy efficiency simulation environment built on POX controller. The replication of state-of-the-art approaches has achieved up to 37% energy saving.

An in-band traffic aware energy efficient approach through minimizing the number of active links in multiple controller environment is presented in [56,57]. The approach presents formal solution for the IP formulated problem and and also heuristics. An energy saving of up to 60% is achieved on real world network topology and traffic traces.

The work in [33] integrates energy aware modules in the open source network management platform OpenNaaS. The modules are designed to enable energy monitoring and energy aware routing for different kinds of OpenFlow controllers. Widely known routing algorithms and scheduling algorithms implemented as part of the routing strategies within the framework are evaluated. Simulation results indicate that the priority-based routing leads energy efficiency improvements of 5% to 35% compared to other routing strategies, with no degradation in network performance. The contribution of this work lays on adding the scheduler and combining it with known routing algorithms. The experiment is done on 3 different controllers: OpenDaylight, RYU, and Floodlight.

Node component scope, node scope, and network scope energy capabilities are discussed in [34]. The node components are the hardware components that make up the switch or the host. Examples of node component scope capabilities are the Adaptive Link Rate and Advanced Configurable and Power Interface. These two techniques are Layer 1 energy efficiency capabilities and are also used in the traditional network paradigm. The node scope deals with sleeping the switch in response to low traffic. The IEEE 802.3 az standard which is the Energy Efficient Ethernet is an example of node scope capabilities. The network scope energy saving capabilities introduce a more complicated problem. The major problem of applying energy saving mechanism at once is that their effect cancels. It is better to apply one technique alone

instead of combining them. This work addresses the effect of the cancelling problem by devising a means of consolidating the three capabilities together. Experiments demonstrate energy savings of up to 54% in particular scenarios.

A traffic and resource aware energy saving method is presented in [55]. Unlike most of the related work in energy efficient optimization problems which are linear, the proposed method formulates the problem as non-linear. Instead of considering snapshot of the state of the network to grasp traffic and resources, the system presented introduces the concept of a discrete time. An important question to ask then is how often to pull statistics and solve the optimization problem. The power saving scheme employed is link level. The heuristic algorithm finds the subset of links to be turned off, and it is shown to achieve more than 30% energy saving.

The energy efficiency problem with multiple controllers in the context of 5G network is presented in [58]. Power consumption is reduced by using traffic engineering techniques, IP formulation for both static and dynamic energy aware routing, and also minimizing the number of active links. As compared to the formal solution, the running time of the two heuristics are better but the energy efficiency is sub optimal.

FLOWP [59] attempts to achieve both power reduction and QoS for fat-tree topology. An IP formulation for power efficient flow scheduling and the corresponding heuristics is proposed. Unlike the other approaches, the status of network, and the minimum threshold for the utility of links and switches are used in the formulations. Experiments demonstrated an energy saving of 30% and better QoS as compared to ElasticTree and CARPO [22, 25].

The main challenge with energy efficiency techniques is that they exhibit reduced performance. The trade-off between energy saving and performance is simultaneously addressed and IP formulated based on the link utility intervals in our prior works [16, 17]. Next Shortest Path and Next Maximum Utility heuristics give priority to performance and energy saving, respectively. However, the framework we presented in [16] proposes a single heuristics that achieve both energy saving and performance at the same time by maximizing parameter named the Energy Profit Threshold (EPT).

Experiments conducted on real network topology and traces provided by SNDLib show an energy saving of up to 50%.

IP formulation for energy aware routing in carrier-grade Ethernet networks with the objective of turning off links and switches is presented in [60]. A greedy bin packing problem based heuristics named first-fit is presented as a solution. Results show that the first-fit heuristics is scalable and achieves an energy saving of 37%. Unlike other approaches, rule space in the switches is considered as a constraint. The approach is tested using SNDLib real world network topology and traffic traces.

Multiple Topology Switching with Data Plane Forwarding Path Rerouting (MTSDPFPR) is an energy saving approach based on sleeping and rate adaptation of links [61]. MTSDPFPR is a multi-level dynamic topology switching mechanism thorough finding the maximum utilized link. The approach is tested on real world GEANT network topology and traffic traces. Experimental results show that energy saving of 25% is achieved.

The application scenarios of the traffic aware solutions are generally on campus networks or data centers. However, the specific techniques have switch, port, link, or network wide applicability. The methods that use queue engineering ([27, 28, 30, 34, 65]) for instance are focused on the switch level queues but their general application is data center or campus networks. The application scenarios in [22, 23, 25, 26, 31, 33] are data centers. On the other hand, the methods in [53, 56, 57] are applicable to a general network environments, either specific network topologies or general network topologies.

Chapter 4

End System Aware Energy Saving Optimization Model and Methods in SDN

4.1 Overview

A data center consists of interconnected servers (physical hosts) that are structured into racks. The end system connected can be a physical machine or a virtual machine. Server virtualization enables running multiple virtual machines (VM) on a single hardware resource and satisfying the applications' resource demands. Resources include CPU, memory, and network bandwidth. VM migration refers to moving a virtual machine from one host to another for the purpose of achieving energy saving, performance increase, load balancing or system maintenance.

Server consolidation refers to a method used in data centers to minimize the number of active servers by increasing the utility of each server [66]. Hence, instead of operating many servers at low utilization, virtualization technique combines the processing power onto fewer physical servers that operate at a higher total utilization. The deployment of SDN in cloud data center virtual machines boosts QoS and load balancing due to the enhanced flexible control of the network.

Network optimization deals with the energy consumption and communication cost of network components. Focusing only on server consolidation may lead to low network performance. Besides, the network components incur a substantial amount of energy. SDN provides global information about the network, such as the topology, bandwidth utilization, physical machine status, virtual machine status, workload, and other performance statistics. It also provides a flexible way to install forwarding rules so that simultaneous migration of virtual machines can be handled.

Unlike traffic aware techniques, where the network components are the focus for energy saving, end system aware techniques consider both network components and end systems as sources of energy saving. Devising an efficient solution needs periodic updates on the status and organization of the end systems and the traffic at the same time.

4.2 End System Aware Model

The goal of the end system aware solution is to minimize the number of active physical machines through migrating the virtual machines into fewer number of physical machines. Reviewing various models used to solve the problem, we propose a general model for the end system aware energy efficiency. End system aware solutions should be addressed as server consolidation and network optimization problems simultaneously [35, 36, 67]. The general model we propose addresses both problems and is inspired by [36].

The problem of virtual machine migration is modeled as a quintuple ($\mathbb{P}$, $\mathbb{V}$, $\mathbb{R}p$, $\mathbb{R}v$, $\mathbb{R}$) where $\mathbb{P}$, $\mathbb{V}$, $\mathbb{R}p$, $\mathbb{R}v$, and $\mathbb{R}$ correspond to the set of physical machines, set of virtual machines, matrix of resources of physical machines, matrix of resource requirements of virtual machines, and vector of type of resources, respectively. The resources are listed as but not limited to CPU, memory and bandwidth capacities. The dimensions of the matrix $\mathbb{R}p$ and $\mathbb{R}v$ are $|\mathbb{P}|$ by $|\mathbb{R}|$ and $|\mathbb{V}|$ by $|\mathbb{R}|$, respectively.

Let X be a placement matrix where

$$X_{ij} = \begin{cases} 1, & \text{if virtual machine} j \text{ is placed on physical machine } i \\ 0, & \text{otherwise} \end{cases}$$

P_i^r is the amount of resource type r available on physical machine i, whereas V_j^r is the demand of virtual machine j for resource type r. Let PM_i is a binary variable with value 1 if physical machine i is on, or 0 otherwise.

$$\text{minimize} \sum_i PM_i \tag{4.1}$$

$$\text{subject to} \sum_{j=1} V_j^r * X_{ij} \leq P_i^r \quad \forall i, r \tag{4.2}$$

$$\sum_i X_{ij} = 1 \quad \forall j \tag{4.3}$$

$$PM_i \geq X_{ij} \quad \forall i, j \tag{4.4}$$

where $1 \leq i \leq |\mathbb{P}|$, $1 \leq j \leq |\mathbb{V}|$ and $r \in \mathbb{R}$.

The first objective function (equation 4.1) minimizes the number of physical machines turned on. The constraint 4.2 states the sum of resource demands of virtual machines installed on a given physical machine cannot be more than the capacity of the physical machine. Constraint 4.3 limits that each virtual machine can be placed on exactly one physical machine. Equation 4.4 associates the variables PM_i and X_{ij} by asserting that a physical machine will be turned on or off depending on whether it is used or not.

The second objective that needs to be considered is the network energy consumption. Let $\mathbb{Q}$ be a traffic matrix where q_{ij} is the amount of traffic between VM_i and VM_j, and b_{ij} is the number of switches that the traffic between physical machines hosting virtual machines i and j traverses.

$$\text{minimize} \sum_{i} \sum_{j} q_{ij} * b_{ij} \tag{4.5}$$

$$\text{subject to} \sum_{j=1} V_j^r * X_{ij} \leq P_i^r \quad \forall i, r \tag{4.6}$$

$$\sum_{i} X_{ij} = 1 \quad \forall j \tag{4.7}$$

The objective function of the general end system aware solution is the combination of the server consolidation and network optimization. Equation 4.5 minimizes the communication traffic between virtual machines and the number of switches traversed between physical machines. Equation 4.6 constrains the model to respect the sum of the virtual machine resources not to exceed the resources of the physical machine that they are placed on. Equation 4.7 asserts that a virtual machine can be placed on exactly one physical machine.

4.3 Methods

Table 4.1 shows the summary of end system aware solutions to energy saving in data centers. The environment column presents the experimental test environment (Testbed, ns-2, or CloudSim) used. The network optimization column shows if an approach considers optimization of the communication cost among virtual machines. Dynamicity is the ability to do server consolidation and network optimization real time. The topology column describes if the method is explicitly applied to a specific topology. The '-' symbol in the table means unknown for the environment and topology columns, or does not apply for the network optimization and dynamicity columns. As shown in Table 4.1, the majority of the end system aware solutions focus on network optimization in addition to virtual machine placement. Only a few approaches consider online virtual machine migration and traffic consolidation. Unlike traffic aware energy efficient methods, most solutions are topology specific mainly fat-tree topo-

logy. Similar to the traffic aware energy efficient techniques in SDN, experiments are conducted on either special testbeds, or simulation and emulation environments.

Table 4.1: End System Aware Techniques for Energy Efficiency

Approach	**Environment**	**Network Optimization**	**Dynamicity**	**Topology**
Honeyguide [35]	Testbed	✓	-	Fat-tree
JointVM Placement [36]	Testbed	✓	✓	Fat-tree
Joint Host VM [37]	ns-2	✓	✓	Fat-tree
Joint Host VMP [39]	Testbed	✓	-	Fat-tree
EQVMP [38]	ns-2	-	-	-
PowerNets [39]	Testbed	✓	-	Fat-tree
MTAD [40]	CloudSim	✓	✓	-
Load Balancing [68]	Testbed	✓	✓	-
QRVE [69]	Mininet and OpenDaylight	✓		-
naVM [70]	Testbed	✓	✓	-
Int Load Balancing [71]	Testbed	✓	✓	-
CloudDC [72]	Testbed	✓	✓	Fat-tree
SLA-Based [73]	Testbed	✓	✓	-
loadbalncing [74]	Testbed	✓	✓	-

Virtual machine placement plays a major role in energy savings of data centers. The work in [75] presents a holistic model for data center power consumption minimization. The model is holistic in the sense it considers all energy incurring components including cooling and virtual machine placement. According to [67], a virtual machine placement method should take into account resource constraints such as physical server and network link capacities.

Honeyguide is a virtual machine migration-aware extension of the fat-tree topology network topology for energy efficiency in data center networks [35]. Reducing energy consumption is achieved by decreasing the number of active (turned on) networking switches. In this approach, the focus is not only turning off inactive switches,

but also trying to maximize the number of inactive switches. To increase the number of inactive switches, two techniques are combined: virtual machine and traffic consolidation. As an extension of existing fat-tree topology, Honeyguide adds bypass links between the upper-tier switches and physical machines. By doing so, it meets the fault tolerance requirement of data centers. Experimental results show relative increase in energy saving of up to 7.8%.

A joint virtual machine placement and routing solution is presented in [36]. Dynamic virtual machine placement and routing is modeled as a combinatorial problem. A Markov approximation is used to solve the joint optimization problem under varying workloads. The method used shows better performance as compared to the common heuristic techniques. Unlike other methods ([35, 38, 39]), this work formulates the workload distribution among virtual machines separately.

An OpenFlow based joint virtual machine placement and flow routing optimization is designed [37]. Unlike the work of [36] the problem is formulated as an integer linear program and applies a series of techniques to remodel and combine the multi-objective problem into a single objective function. The simulation results show that this approach performs up to 67% energy saving, and it performs better than the existing server consolidation only or network optimization only solutions.

EQVMP proposes energy-efficient and QoS-aware virtual machine placement for SDN based data centers [38]. Unlike ElasticTree, power on and off is applied to the servers themselves instead of the switches, the focus is server consolidation. EQVMP combines three techniques: hop reduction, energy saving, and load balancing. Hop reduction divides the VMs into groups and reduces the traffic load among groups by graph partitioning. Energy savings mostly are achieved by VM placement. The motivation behind VM placement is from Best Fit Decreasing and Max-Min Multidimensional Stochastic Bin Packing. Fat-tree is used to represent the VM and servers in the data center. SDN controller is used to balancing the load in the network. Load balancing achieves flow transmission in networks without congestion. Experimental results show an energy saving of 25%.

PowerNets [39] is a joint optimization technique for both server consolidation and network optimization. The peak hours of servers in a data center are not the same. Therefore, the motivation of this work is based on the observation that all the servers do not use their maximum capacity simultaneously. Correlation on the workloads that run on the servers gives a way of grouping servers on their time series. Workloads that use similar servers in a close time interval are grouped to use the same path. By leveraging from server consolidation, DCN optimization and correlation of workloads, PowerNets provides up to 51% energy saving. Different than our general model, PowerNets takes the correlation of workloads among machines into consideration.

A multiobjective virtual machine placement approximation method is presented in [40]. The motivation is that virtual machine placement algorithms for cloud data centers find local solutions for each objective and finally combine the results. Such approaches fail to guarantee a global optimum. The multiobjective approximation model hence combines placement efficiency, load balancing, resource utility, and energy efficiency objectives into one. A multi-objective heuristic algorithm for virtual machine placement named MTAD is proposed. MTAD is based on a greedy algorithm, minimum cut and best-fit algorithms. Experiments are conducted on simulation environment and have shown better results than greedy, simulated annealing, and genetic algorithms in the literature. MTAD considers server consolidation, network optimization and also operates online to accommodate dynamic changes.

A load balancing energy-saving approach for geographically distributed data center networks is proposed in [68]. A multi-objective IP formulation that both balances request load and minimizes brown energy consumption by putting hosts to sleep. Experiments conducted using real-world traffic traces exhibit energy savings of up to 42%.

The joint problem of power consumption of data center network, SLA violation, QoS-aware, and VM placement are simultaneously addressed in [69]. IP formulation and elephant flow detection based heuristics are presented. Unlike other approaches, a distributed SDN architecture is considered. The experiments are conducted on Min-

inet emulation environment and OpenDaylight controller achieve 21% energy saving.

A taxonomy for virtual machine placement solutions is presented in [42]. It also presents metrics to classify virtual machine placement methods by elasticity, over-booking and time awareness. Elasticity is the responsiveness of the system to saturation and under-utility of resources. Server or network overbooking is the re-utilization of resources when they are idle. Since a workload or a service uses a given service for finite length of time, update on the reusability of resources is important.

A variation of multi-commodity minimum cost flow problem is used to model virtual machine migration and load balancing. The two constraints are load balancing and migration time. Whereas the former attempts to achieve virtual machines are evenly distributed to the physical machines, the latter puts a time constraint to do the actual migration. Ant colony heuristics algorithm is provided as a solution. Experimental results show that the migration time is decreased by half as compared to similar approaches. Although this work is not directly related to energy saving, the formulation of the ant colony heuristics can further be studied for efficiency purpose [70].

A joint optimization effort of energy efficiency and virtual machine migration in a cloud data center is presented in [72], and a similar approach is also proposed in [71]. Viewing the virtual machine placement and migration problem as a Multidimensional Vector Bin Packing Problem (MVBPP) is one of the common approaches. However, with the increase in the dimension of the vector, most heuristics fail to find optimal results. In this approach, a mathematical formulation and a hybrid single-parent (partheno-genetic) algorithm to solve the virtual migration integration problem is proposed. Simulation results show that the approach exhibits less migration time as compared to similar approaches. However, it does not fully solve the problem of scalability.

A comparison of several meta-heuristics algorithms used in SDN is presented in [74], where different metrics to evaluate meta-heuristics algorithms used for load balancing in SDN are suggested. The metrics are throughput, migration time, energy

efficiency, and online migration. Some examples of meta-heuristics algorithms are Genetic algorithm, Ant colony optimization, Particle swarm optimization, Greedy, and Simulated annealing. The major reason to resort to such algorithms is due to the fact that the formal solution is not time efficient and converges for a small number of physical and virtual machines. On the other hand, the meta-heuristics algorithms provide sub-optimal solutions and in some cases, they are far from optimal.

A Service Level Agreement (SLA) aware resource overbooking for energy efficient workload allocation and virtual machine placement is presented in [73]. The approach solves the fixed amount of resource allocation using SDN to dynamically consolidate traffic and control of Quality of Service (QoS) from a logically central point. It jointly minimizes the power consumption of the host and the network and the number of SLA violations.

The application scenarios of the end system aware solutions are SDN based data centers, either centralized or geographically distributed. The energy saving objective focuses on minimizing the number of active physical machines through migrating virtual machines. Moreover, balancing the workload on virtual machines is also another focus that needs to be considered. The end system aware methods we presented in this work are applicable to scenarios where there are virtual machines and workloads that can be assigned and migrated to different machines.

Chapter 5

Framework for Traffic Proportional Energy Efficiency in SDN

Traffic proportional energy efficiency is an attempt to make energy cost proportional to the amount of traffic streaming through the network. A practical solution for such problem is to sleep/turn off under-utilized components for low traffic volume. Minimizing the components turned on to accommodate a given traffic, however, degrades performance [17, 22, 25, 32–34]. Designing energy efficient solutions is non-trivial since they need to tackle the trade-off between energy efficiency and network performance.

In a dynamically changing network environment, the practicality of such solutions in traditional networking is impossible. However, SDN enables the network to adapt the changing environment by recomputing paths, optimizing routes and increasing flexibility. The controller periodically gathers statistical information about the traffic, the status of the switches, the status of the links, the status of the end systems, and the topology of the network. Related works in energy aware routing are focused on minimizing the number of active network components [22, 23, 25, 31, 53]. In this chapter, we propose a three-component framework that makes energy consumption proportional to the traffic volume in a dynamic environment. The contributions of this chapter are as follows

- We propose a traffic proportional energy efficient framework for SDN. The components in the framework deal with traffic monitoring, topology management, and optimization.

- We also propose two energy efficient topology structure independent heuristics with focus of network performance and utility independently.

The remainder of the chapter is organized as follows. Section 5.1 describes traffic proportional framework we proposed and two example heuristics to solve the for minimizing the energy cost of network components. Section 5.3 concludes and gives background for chapter 6.

5.1 Traffic Proportional Framework

The goal of the traffic proportional framework is to design a controller which makes the energy consumption of the network proportional to the traffic volume. Figure 5.1 illustrates the traffic aware module of the framework. The information of the traffic generated from the applications running on the top of the controller are passed to the Traffic Manager. The Traffic Manager passes this traffic information to the TA Optimizer. The statistics of the network and the topology information is fed to the TA Optimizer from the switches. The Optimizer generates an optimal subgraph based on the traffic volume and the utility of the links. Since the framework is designed to work in a dynamic environment, a low traffic load should result in a subgraph with smaller number of active links and switches, whereas high traffic load should increase the number of active links and switches in the subgraph respectively. In the former, the sleep/turn-off decision for a subset links and switches would be achieved. In the latter, however, a subset of links and switches would be turned on or made active.

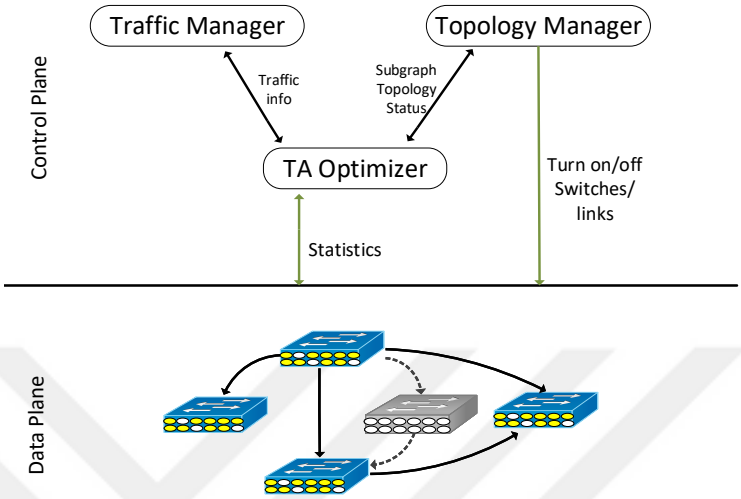


Figure 5.1: A three-component framework for traffic proportional energy efficiency in SDN

5.2 Algorithms for Energy Aware Routing

We propose topology independent heuristic algorithms based on the utility of links, that are Next Shortest Path (NSP) and Next Maximum Utility (NMU) which correspond to the replacing the under-utilized links with next shortest path and with the next path in the direction of the maximum link utility, respectively. Algorithm 1

Algorithm 1 ShortestPathInitialization($\mathbb{G}$, $\mathbb{F}$)

1: $\mathbb{U} \leftarrow Initialize(0)$
2: **for all** $f \in F$ **do**
3: $\quad path_f \leftarrow shortestpath(sr, ds, \lambda_f)$
4: $\quad$ **for all** $e_{ab} \in path_f$ **do**
5: $\quad\quad U_{ab} \leftarrow U_{ab} + \dfrac{\lambda_f}{W_{ij}}$
6: $\quad$ **end for**
7: **end for**
8: **return** $\mathbb{U}$

initializes utilities of the links using shortest path algorithm for each flow. It takes the graph $\mathbb{G}$, which represents the network topology and traffic demand $\mathbb{F}$. For a given link e_{ij}, if a traffic flow $f = (sr, ds, \lambda_f)$ passes through it, its corresponding utility U_{ij}

is incremented by $\frac{\lambda_f}{W_{ij}}$. The algorithm returns the set $\mathbb{U}$ that includes the utilities of all links. Algorithm 2 takes the utility set $\mathbb{U}$ and threshold $Umin$, and returns the list of links with utilities less than or equal to the threshold.

Algorithm 2 ReturnCandidateList($\mathbb{U}$, $Umin$)

```
1: CandidateList ← []
2: for all U_ij ∈ 𝕌 do
3:     if U_ij ≤ Umin then
4:         CandidateList.append(e_ij)
5:     end if
6: end for
7: return CandidateList
```

Algorithm 3 NSP: NextShortestPath($\mathbb{G}$, $\mathbb{F}$, $Umin$)

```
1: 𝕌 ← ShortestPathInitialization(𝔾, 𝔽)
2: CandidateList ← ReturnCandidateList(𝕌, Umin)
3: for all e_ij ∈ CandidateList do
4:     Path_ij ← shortestpaths_ij
5:     SP ← Path_ij[0]                          ▷ Pick one of the shortest paths
6:     for all f passing through e_ij do
7:         for all e_ab ∈ SP do
8:             U_ab ← U_ab + λ_f / W_ab          ▷ increment U_ab
9:         end for
10:        U_ij ← U_ij − λ_f / W_ij              ▷ decrement U_ij
11:    end for
12:    if U_ij == 0 then
13:        L_ij ← 0                             ▷ state of link is inactive
14:        𝔾 ← Turn off(e_ij)
15:    end if
16: end for
17: return 𝔾                                    ▷ Subgraph
```

Algorithm 3 (NSM) aims at identifying under-utilized links and re-routes traffic flows passing through them to the next shortest alternative path. The algorithm takes topology $\mathbb{G}$, set of flows $\mathbb{F}$ and threshold $Umin$ as inputs. It first initializes the link utilities based on the shortest path algorithm. It then selects the under-utilized

links (invoking algorithm 2) and sets the $CandidateList$. The $shortestpaths_{ij}$ (line 4) returns the list of paths from Z_i to Z_j excluding those links in the $CandidateList$. The paths stored in $shortestpaths_{ij}$ are sorted according to the path length. The $shortestpaths_{ij}$ is set to $Path_{ij}$ (line 5). NSM algorithm then picks one of the shortest paths SP to replace the direct link. Replacing a direct link with alternative path needs redirecting the flows passing through it. Basically, we need to increase the utilities of the hops composing SP and decrement the utility of the direct link (lines 8 and10). The status of a link is updated if the utility is 0 (all the flows passing are redirected to alternative path SP), hence updates the graph $\mathbb{G}$'s status (lines 12-15). Algorithm 3 returns the updated sub graph of $\mathbb{G}$.

Algorithm 4 (NMU) aims at replacing the candidate links with the path that has the link with maximum utility. After identifying the candidate links (line 2), list of replacement paths except for links in $CandidateLis$ $minutilitypath_{ij}$ is assigned to $Path_{ij}$ (line 4). $Path_{ij}$ is sorted according to maximum utility link. UP is the first path in the $Path_{ij}$. The algorithm then updates the utilities of links and the graph $\mathbb{G}$ in the remaining part with similar procedure of Algorithm 3.

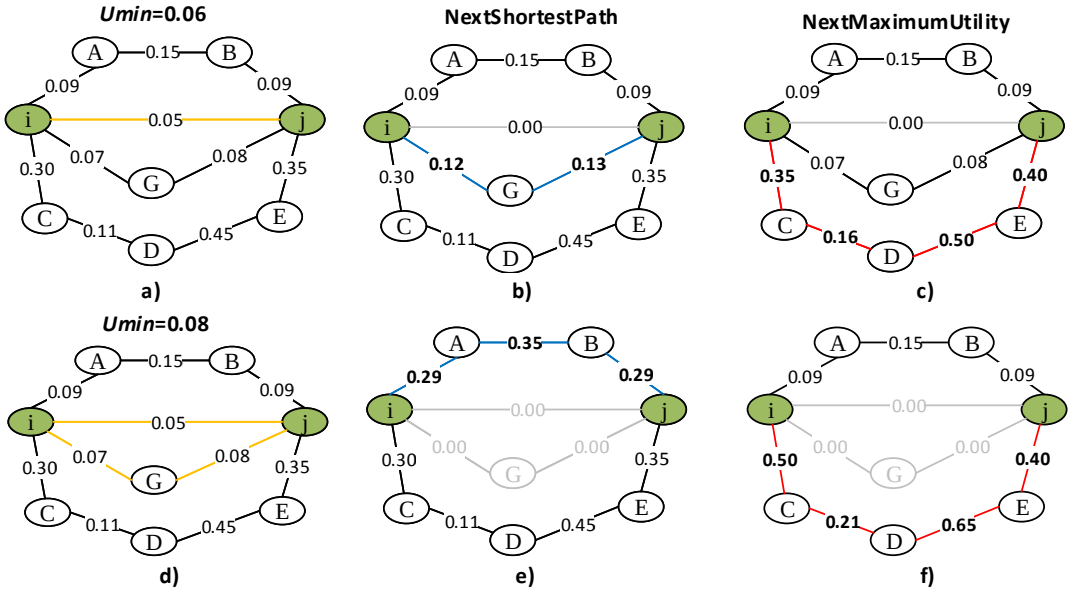


Figure 5.2: Examples for NSP and NMU utility based heuristics

Algorithm 4 NMU: NextMaximumUtility($\mathbb{G}$, $\mathbb{F}$, $Umin$)

$\mathbb{U} \leftarrow ShortestPathInitialization(\mathbb{G}, \mathbb{F})$
$CandidateList \leftarrow ReturnCandidateList(\mathbb{U}, Umin)$
for all $e_{ij} \in CandidateList$ **do**
    $Path_{ij} \leftarrow maxutilitypaths_{ij}$
    $UP \leftarrow Path_{ij}[0]$
    for all f `passing through` e_{ij} **do**
        for all $e_{ab} \in UP$ **do**
            $U_{ab} \leftarrow U_{ab} + \frac{\lambda_f}{W_{ab}}$
        end for
        $U_{ij} \leftarrow U_{ij} - \frac{\lambda_f}{W_{ij}}$
    end for
    if $U_{ij} == 0$ **then**
        $L_{ij} \leftarrow 0$ ▷ state of link is inactive
        $\mathbb{G} \leftarrow$ `Turn off`(e_{ij})
    end if
end for
return $\mathbb{G}$

Figure 5.2 shows an example of how the algorithms work. As given in Fig5.2.a, utility of the link connecting nodes i and j is 0.05 and the minimum ratio to make a link active $Umin$=0.06. Thus, link e_{ij} is a candidate link to be turned off. $Path_{ij} = [i, G, j], [i, A, B, j], [i, C, D, E, j]$ where number of hops in each path is 2, 3, and 4 respectively. $SP = [i, G, j]$ and the number of hops is 2. According to NSP (Algorithm 3), the replacement path for the direct link, as illustrated in Fig.5.2.a,is to pick the next shortest path among $Path_{ij}$ which is $[i, G, j]$. The NMU algorithm on the other hand, selects the path which contains the link with the maximum utility $[i, C, D, E, j]$ with a path length of 4 hops. After the updates, the flows passing through the edge e_{ij} would be redirected to the replacement path. Assuming all links have equal bandwidth in this example, the utility ratio of the link would be added to each link of the replacing path. The outcomes of NSP and NMU algorithms applied to Fig.5.2.a are illustrated in Fig.5.2.b and c, respectively.

The minimum threshold value $Umin$ directly affects the set of candidate links. As another example, in Fig.5.2.c, the threshold $Umin$ is set to 0.08. The candidate links

in this case are $e_{i,j}$, $e_{i,G}$ and $e_{G,j}$. The results of the NSP and NMU algorithms are shown in Fig.5.2.d and e, respectively.

5.3 Conclusions

Energy efficiency has become a critical concern for IT equipment designs. The main problem in networks, however, is energy consumption is constant regardless of the volume of traffic. The flexible control provided by SDN paved the way to optimize the efficiency of network environment dynamically. In this chapter, we presented there module traffic-aware energy efficiency framework for SDN and a heuristics algorithm. The framework is implemented on top of POX controller.

The NSP and NMU heuristics we have proposed give priority to performance and utilities respectively. They are topology independent and can be combined with other heuristics to give better energy-saving or network performance. Besides they work independently, either we have to choose performance or maximum utility not both at the same time as we analyzed in [17]. In the next chapter, we present an energy efficiency metric that is specific to SDN. By maximizing this metric, our objective is to maintain the trade-off between performance and energy efficiency.

Chapter 6

RESDN: A Novel Metric and Method for Energy Efficient Routing in SDN

6.1 Introduction

In data center networks, components are utilized 30% to 40% most of the time [23,76]. The overall power consumption however remains almost the same for varying amount of traffic volume. A practical solution for a traffic proportional energy consumption problem is to sleep/turn off under-utilized components for low traffic volume. Minimizing the number of network components for low traffic, leads to network performance degradation [22, 25, 26, 32–34].

Energy saving at the cost of network performance degradation, however, is an undesirable result. The trade-off between energy saving and performance is quite a challenging task. Energy efficient solutions mainly focus on minimizing the number of energy consuming devices whereas network performance optimization solutions mainly focus on keeping the devices work at full capacity regardless of low traffic. The two opposing objectives clearly signifies the need for a holistic solution that maintains the trade-off between efficiency and performance.

In SDN, a naive way for measuring the efficiency of a network is to take the ratio of the power used to the maximum power consumption of the network components. A wide range of energy efficiency metrics like Power Usage Efficiency (PUE) have been proposed and used to measure the efficiency of data centers [77, 78]. The Green Grid and Standard Performance Evaluation Corporation (SPEC) are the most widely known efforts among the many. The metrics proposed, however, are either applicable to data centers only or do not consider the utilities of the network resources, and cannot directly be applied to dynamically changing and programmable networks.

Therefore, addressing the need for a metric to measure the energy efficiency of a network with regard to traffic volume and utility of resources, we propose a novel energy efficiency metric, namely RESDN (Ratio for Energy Saving in SDN), that quantifies energy efficiency based on utility intervals defined by minimum and maximum link utility parameters.

The contributions of this chapter are as follows.

- We propose a novel energy efficiency metric RESDN: the ratio for energy saving in SDN that quantifies energy efficiency based on link utility intervals. To the best of our knowledge, the approach is unique as it measures how links are profitably utilized in terms of the amount of energy they consume with respect to their utility.

- We develop Integer Programming (IP) formulation with the objective of maximizing RESDN and propose heuristics algorithm MaxRESDN for achieving the objective.

- We conduct extensive experiments on Mininet network emulator and POX controller using real network traffic traces and present comparative quantitative analysis of MaxRESDN method. The performance metrics of interest are switch power consumption, RESDN value, percentage of links saved, average path length, throughput, delay, and traffic proportionality.

- We simulate the power consumption of two hardware switches (NEC and Zodiac FX) and one virtual switch (Open vSwitch-OvS) that are OpenFlow enabled, and conduct comprehensive experiments to measure the power consumption of our proposed MaxRESDN heuristic algorithm.

- MaxRESDN heuristic algorithm achieves the highest ratio for energy saving which is up to 30% better than similar state-of-the-art utility-based heuristics algorithms. It also saves up to 38% links and exhibits an average path length closer to the best energy saving heuristics that focuses on performance. MaxRESDN has the highest traffic proportional energy consumption which 3 to 5% better than similar utility-based heuristics. To the best of our knowledge, our approach is the first in maintaining the trade-off between energy efficiency, network performance, and traffic proportionality by optimizing RESDN metric.

- Switch power consumption results show that MaxRESDN exhibits on average up to 14.7 watts, 10 watts, and 3.2 watts less power consumption for NEC, OVS and Zodiac FX switches respectively as compared to other utility-based heuristics for energy efficient routing.

- As the utility parameters directly impact the performance of the MaxRESDN heuristics, we conduct a detailed analysis of the parameters with respect to a range of traffic volumes.

The remainder of the chapter is organized as follows. Section 6.2 presents related work on energy efficiency in SDN and energy efficiency metrics. Section 6.3 presents preliminary work on utility based traffic proportional energy saving in SDN. Section 6.4 describes RESDN metric, provides IP formulation for RESDN maximization, and presents the MaxRESDN heuristics method that maximizes the RESDN of a network.

6.2 Related Work

The control and forwarding plane separation in SDN has led to flexibility for many network services ranging from load balancing, dynamic routing, energy efficient and flexible network control [1]. The primary challenge of energy efficiency in networking is the fact that energy consumption is not proportional to the volume of traffic. The problem is even more difficult in the traditional networking since there is a very limited flexibility. Traffic-aware energy efficient routing techniques in SDN attempt to make energy consumption proportional to the traffic volume. We discuss related work in traffic-aware energy efficient routing in subsection 6.2.1. Measuring the energy efficiency of a network environment including data centers have been investigated and various metrics were also proposed. In subsection 6.2.2, we discuss related work in this area and why we needed a new metrics for SDN.

6.2.1 Traffic Aware Energy Efficient Routing Techniques

Traffic aware energy efficient routing techniques in SDN can be classified based on the energy saving capabilities they focus on and the topology structure they are designed for. The main energy saving capabilities are the links and the switches. Some of the works [25,26,32–34] focus on links, where some others focus on forwarding switches [24, 30]. There also exist works [17, 22, 53] that consider the combination of links and switches as the energy saving components. Furthermore, queue engineering based techniques consider the arrival of packets and their waiting times to decide per-port power requirement [30–34, 79, 80].

Based on the topology assumption, energy efficient methods can be classified as general or tailored to a specific network topology. Some methods are general in the sense that they are applicable to any kind of network topology [17, 53, 64] and some are only applicable to specific topologies such as Fat-tree, butterfly, and BCube, where Fat-tree and BCube topologies are among the widely used structures used in organizing end systems in data centers [22, 23, 26, 27].

Table 6.1 presents the summary of traffic aware energy efficient techniques in SDN. The Topology column shows the kind of topology structure the approach is designed for and tested on. The Utility-Based column indicates if the approach focuses on the utility of the links. The Link column shows if the method considers links as energy saving component. Queue Engineering column shows if the approach applies queue engineering techniques.

Table 6.1: Classification of Energy Efficient Routing Techniques in SDN

Approach	**Topology**	**Utility Based**	**Link**	**Queue Engineering**
ElasticTree [22]	Fat-tree	-	✓	-
Carrier Grade [23]	General	-	✓	-
CARPO [25]	Fat-tree	-	✓	-
EnableOpenflow [26]	Fat-tree	-	✓	-
Traffic distribution [81]	General	✓	✓	-
GreenSDN [32]	Bcubic & Fat-tree	-	✓	✓
Fine-grained [82]	General	✓	✓	-
OpenNaas [33]	-	-	✓	✓
Orchestrate [34]	-	-	✓	✓
Utility-based [17]	General	✓	✓	-
Dynamic TA [53]	General	-	✓	-
REsPoNse [24]	-	-	-	-
NetFPGA QE [30]	-	-	-	✓
GreenRE [31]	General	-	✓	-
Bandwidth-aware [79]	Bcubic & Fat-tree	- ✓	-	✓
RA-TAH [27]	Fat-tree	-	✓	✓
TE based [29]	General	-	✓	-
Re-Routing [28]	General	-	-	✓
Resource-aware [55]	Fat-tree	-	✓	-
SDN /Ethernet [60]	General	-	✓	-

Energy efficient techniques use a method of sleeping or turning of unused switches or links when the traffic volume is low and turn them on when traffic volume increases [26, 26, 79, 83]. The objective of energy efficiency is reducing the number of active network components. However, this has an adverse effect on the performance of the network hence degrades the network performance. There is a trade-off between

the two opposite objectives which are minimizing energy consumption and increasing performance.

A closely related work considers the remaining bandwidth of links [79]. After formulating the problem using IP, they propose a scheduling algorithm. The heuristics proposed schedules paths for the new flows by considering the paths followed by the preceding flows. Unlike this approach, our focus is not on the remaining bandwidth of the links but the utilities of the links. We measure how much each link is utilized based on the utility interval defined by the minimum and maximum utility parameters.

In contrast to the approach of turning off links for energy saving [17, 25, 31, 84], the work [81] examines if turning off / sleeping switches always result in energy saving. The network is modeled as a graph with the nodes as the switches and the edges as the links. Comprehensive experiments show for cases where the routing power consumption is cubic with respect to traffic volume, distributing the traffic to underutilized links exhibit a better energy saving as compared to turning them off. Since the routing power consumption is polynomial in general, distributing traffic over underutilized links is only applicable to specific cases [81].

The other approach close to our work is [53] which formulates the energy efficiency problem with mixed integer programming (MIP), then proposes heuristic algorithms. There are four variations of the heuristics where two of which namely Shortest Path First (SPF) and Shortest Path Last (SPL) sort the flows in the order of the shortest path. The other two heuristics namely Smallest Demand First (SDF) and Highest Demand First (HDF) sorts the flows according to the rate flow.

In contrast, our approach is flow order independent, it focuses not only on energy efficiency but also in performance, considers the utility of each link, and proposes a link utility-based energy efficiency metric that simultaneously quantifies efficiency and performance.

6.2.2 Energy Efficiency Metrics

Data center energy efficiency metrics have been studied widely with companies and standardization organizations. Some of these are the Green Grid, Standard Performance Evaluation Corporation (SPEC), and Transaction Processing Performance Council (TPC) [85, 86]. The Green grid consortium proposes several metrics to measure infrastructure power consumption, carbon dioxide emission, water usage efficiency,and the rate of useful information processed related to resources used. Power Usage Effectiveness (PUE), Data Center Infrastructure Efficiency (DCIE), Carbon Usage Effectiveness (CUE), Water Usage Effectiveness (WUE), and Data Center Productivity (DCP), all of which measure energy efficiency and sustainability.

Power Usage Effectiveness (PUE), the most prevalent metric, is the ratio of the total annual amount of energy that comes into a data center to the energy that is used by IT equipment [87]. IT equipment includes workstations, storage, monitors, and network devices. Total data center energy consists of IT equipment energy consumption plus energy for cooling system components, UPS, switch gear, and data center lightening. The ideal value for PUE is 1 which means all the energy in the data center is used by the IT equipment. The main limitations of PUE metric is that it is only applicable to a single building that supports a data center [84]. The other limitation of PUE is that it does not focus on specifically on links and switches which are the main energy saving capabilities in SDN.

Data Center Infrastructure Efficiency (DCIE), which is the inverse of PUE, is calculated as the ratio of IT equipment power consumption to total facility power consumption. Carbon Usage Effectiveness (CUE) is the product of the amount of carbon dioxide emitted per kilowatt hour (CEF) and the data center's annual PUE [88]. Water Usage Effectiveness (WUE) is the ratio of the annual site water usage in liters to the IT equipment energy usage in kilowatt hours (Kwh) [89]. WUE measures the water consumption in relation to IT equipment energy consumption. However, it is only applicable to a single data center site and does not consider network devices which are the main focus of energy saving in SDN. Data Center Productivity

(DCP) measures the quantity of useful information processing completed relative to the amount of some resource consumed in producing the work [77]. DCP treats the data center as a black box where an power enters in to the data center do some useful task and leaves. An important concern in this is how to quantify a useful work.

To the best of our knowledge, there is no metric that is fully focused on network components such as switches and links which are energy saving capabilities in SDN. Using the existing metrics in an SDN environment falls short of capturing the utilities of links which is an important energy saving capability of a network environment. The other drawbacks in the metrics are that they are measured annually and offline. It makes the necessity for new metrics that can measure dynamically changing network environment an indisputable argument. Failing to do so undermines the flexible network management capability that could have been exploited from SDN.

6.3 Preliminaries

Next Shortest Path (NSP) and Next Maximum Utility (NMU) heuristics were proposed in our prior work [17] that are generally applicable to any network topology and are flow order independent. NSP and NMU first select those links with utility less than $Umin$ as the candidate links, then aim at redirecting flows passing through the candidate links to the next shortest path or the path in the direction of the link with maximum utility, respectively. The rationale behind selecting links with utilities less than $Umin$ as a candidate is to redirect flows that pass through these links to make them inactive. NSP and NMU attempt to put underutilized links to inactive state. By doing so not only does the number of active links is reduced, but also overall utilities of the links increase. Next, we discuss the NSP algorithm in detail as the preliminary, since it attempts to make every link to be utilized above the minimum utility $Umin$ or to be made inactive.

NSP (Algorithm 5) identifies under-utilized links and re-routes traffic flows passing through them to the next shortest alternative path. The algorithm takes topology as

a graph $\mathbb{G}$, set of flows $\mathbb{F}$, values $Umin$, and $Umax$ as inputs. It first initializes the link utilities by taking the current utilities of the links, and then selects the under-utilized links and sets the $CandidateList$. Line 6 returns the list of paths from Z_i to Z_j excluding those links in the $CandidateList$. The paths stored in $shortestpaths_{ij}$ are sorted according to the path length. The $shortestpaths_{ij}$ is set to $Path_{ij}$ (line 7). NSP algorithm then picks one of the shortest paths SP to replace the direct link. SP is replaced with the next $Path_{ij}$ until all the links in $\forall e_{ab} \in SP$, $U_{ab} + U_{ij} \leq Umax$ (lines 8 to 10). Replacing a direct link with alternative path needs redirecting the flows passing through it. Basically, the utilities of the hops composing SP are increased and the utility of the direct link are decreased (lines 13 and15). The status of a link is updated if the utility is 0 (all the flows passing are redirected to alternative path SP), hence updates the graph $\mathbb{G}$'s status (lines 17-21). Algorithm 5 returns the updated sub graph of $\mathbb{G}$.

Algorithm 5 NSP: NextShortestPath($\mathbb{G}$, $\mathbb{F}$, $Umin$,$Umax$)

Input: Graph $\mathbb{G}$, set of traffic flow $\mathbb{F}$, minimum utility $Umin$, and maximum utility $Umax$
Output: Sub graph of $\mathbb{G}$
$\mathbb{U} \leftarrow NetworkStatus(\mathbb{G}, \mathbb{F})$
$CandidateList = ReturnCandidateList(\mathbb{U}, Umin)$
for all $e_{ij} \in CandidateList$ **do**
    $Path_{ij} \leftarrow shortestpaths_{ij}$
    $SP = Path_{ij}[0]$ ▷ Pick one of the shortest paths
    while $\exists\ e_{ab} \in SP$ where $U_{ab} + U_{ij} > Umax$ **do**
        $SP =$ `next`$(Path_{ij})$
    end while
    for all f `passing through` e_{ij} **do**
        for all $e_{ab} \in SP$ **do**
            $U_{ab} = U_{ab} + \frac{\lambda_f}{W_{ab}}$ ▷ increment U_{ab}
        end for
        $U_{ij} = U_{ij} - \frac{\lambda_f}{W_{ij}}$ ▷ decrement U_{ij}
    end for
    if $U_{ij} == 0$ **then**
        $L_{ij} = 0$ ▷ state of link is inactive
        `Turn off(`e_{ij}
        `Update G`
    end if
end for

6.4 RESDN Metric: Ratio for Energy Saving in SDN

In this section, we propose RESDN energy efficiency metric based on link utility interval defined by the minimum and maximum utility parameters $Umin$ and $Umax$. Then, we present IP formulation and heuristics to maximize the RESDN value of a network in a dynamic environment.

6.4.1 Motivation and Definition

Traffic aware energy saving approaches attempt to turn off/sleep network components by observing the traffic volume. Traffic proportional energy consumption is a more stricter requirement than traffic-awareness. It demands the percentage of power consumption of network components to be proportional to the traffic volume. However, none of the existing energy efficiency metrics discussed in Section 6.2 take the utilities of the network components under consideration with regard to the network traffic volume. Addressing this issue, we propose Ratio for Energy Saving in SDN (RESDN) metric that indicates the percentage of links with utilities within the interval defined by $Umin$ and $Umax$ with respect to the total number of active links.

The RESDN value reflects how the network components are profitably utilized. The current mode of payment for customers in cloud computing is Pay-as-you-go (PAYG) which charges based on the usage of computing resources. Similarly, RESDN enables the network provider to implement Pay-as-you-use (PAYU) in SDN for energy consumption. A simple scenario, in this case, is a network provider aiming to pay the energy cost of a network component only if it is utilized within a utility interval expressed in terms of RESDN. This RESDN policy is expressed by $Umin$ and $Umax$ parameters and are enforced by the controller. The controller decides the links and their corresponding utilities to match the utility interval specified by the provider. The technical solution to this is to redirect the flows passing through underutilized links to other links. Especially in a dynamic environment where the traffic volume increases and decreases, the RESDN value changes, the controller is responsible for keeping the RESDN value at maximum. With this, the energy cost of the network provider would be proportional to the traffic volume being streamed in the network at any time.

The RESDN metric enables to measure the energy efficiency of the network by calculating the ratio of the number of links with utilities between the $Umin$ and $Umax$ to the total number of active links. Thus, it allows the network provider to proactively set the RESDN value.

Table 6.2 shows the parameters used in the formulation of the RESDN . The network is modeled as a weighted graph $\mathbb{G}$= ($\mathbb{Z}$,$\mathbb{E}$) with $\mathbb{Z}$ as the set of switches where $Z_i \in \mathbb{Z}$ represents switch i and $e_{ij} \in \mathbb{E}$ represents that there exists a link between switches Z_i and Z_j. The weight W_{ij} corresponds to the bandwidth of the link e_{ij}.

Table 6.2: Table of Symbols

Parameters	**Description**
$\mathbb{G}$	A graph that consists of $\mathbb{Z}$ and $\mathbb{E}$
$\mathbb{Z}$	Set of switches $Z_i \in \mathbb{Z}$ represents switch i
S_i	Binary variable of switch Z_i
$\mathbb{E}$	Set of links $e_{ij} \in \mathbb{E}$ is a link between switches Z_i and Z_j
L_{ij}	Binary variable of link e_{ij} (active or inactive)
W_{ij}	Bandwidth of the link e_{ij}
$\mathbb{F}$	Set of flows $f \in \mathbb{F} = (sr, ds, \lambda_f)$
f	A flow f=(sr, ds, λ_f) with source , destination and flow rate
F_{ij}	Binary variable of flow f passing through link e_{ij}
$\mathbb{U}$	Set of utilities of all links
U_{ij}	Utility of link e_{ij}
X_{ij}	Binary variable of link e_{ij} $Umin \leq U_{ij} \leq Umax$

Let binary variable S_i denote the status of switch Z_i such that

$$S_i = \begin{cases} 1, & \text{if switch } Z_i \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

Traffic in the network is represented by the set of flows $\mathbb{F}$ where $f \in \mathbb{F}$ is defined as f=(sr, ds, λ_f). sr and $ds \in \mathbb{Z}$ are the source and destination switches of flow f, and λ_f is the rate of f measured in bits per second.

$$F_{ij} = \begin{cases} 1, & \text{if flow } \ f \text{ passes through edge } e_{ij} \\ 0, & \text{otherwise} \end{cases}$$

$\mathbb{U}$ is the set of the utilities of every edge in the graph $\mathbb{G}$ where $U_{ij} \in \mathbb{U}$

$$U_{ij} = \frac{\sum_{\forall f} F_{ij} x \lambda_f}{W_{ij}} \tag{6.1}$$

is defined as the ratio of the sum of the rates of the flows passing through the edge e_{ij} to the link bandwidth W_{ij}. Utility of a link is between 0 and 1, where 0 means no flow is passing through the link and 1 means the sum of the flow rates passing through the link is equal to the link bandwidth. Let $Umin$ be the minimum utility value to keep a link active and $Umax$ is the maximum utility of a link.

$$X_{ij} = \begin{cases} 1, & Umin \leq U_{ij} \leq Umax \\ 0, & \text{otherwise} \end{cases}$$

The energy efficiency metric is formally defined as

$$RESDN = \frac{\sum_{\forall e_{ij}} X_{ij}}{\sum_{\forall e_{ij}} L_{ij}} \tag{6.2}$$

where L_{ij} is binary variable that denotes the status of edge e_{ij}

$$L_{ij} = \begin{cases} 1, & \text{if edge } e_{ij} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

According to Equation 6.2, RESDN value of 1 means that all links that are turned on are operating between the interval defined by parameters $Umin$ and $Umax$. RESDN value of 0 means that none of the links are operating profitably which means that the network is underutilized. The motivation behind the RESDN comes from the idea that network providers want to pay for energy consumption for a given resource only if it is utilized at least to a minimum value of $Umin$. If the utility of a resource is less than $Umin$, then it is underutilized. $Umax$ needs to be less than 1 since setting

it to 1 would create network congestion which in turn degrades performance. The larger the value of RESDN , the more the percentage of the active links that operate between the $Umin$ and $Umax$ values. Smaller RESDN value shows that a large number of active links operating unprofitably.

6.4.2 IP formulation for RESDN Optimization

The objective function of our IP model is to maximize the RESDN value of the network.

$$\text{maximize } RESDN \tag{6.3}$$

$$\text{subject to } \sum_{\forall f} F_{ij} x \lambda_f \leq W_{ij} \ , \forall e_{ij} \tag{6.4}$$

$$\sum_{\forall f} F_{ai} = \sum_{\forall f} F_{ib} \ , \ Z_i \neq f^{sr}, Z_i \neq f^{ds} \tag{6.5}$$

$$F_{mj} = F_{in} \ , \forall f \ Z_m = f^{sr} \ , \ Z_n = f^{ds} \ , \forall e_{mj} \ , \exists e_{in} \tag{6.6}$$

$$F_{ij} \leq S_j \ and \ F_{ji} \leq S_j \ , \forall Z_j \in \mathbb{Z} \tag{6.7}$$

$$S_i \leq \sum_{\forall f} [F_{ij} + F_{ji}] \ , \ \forall Z_i \in \mathbb{Z} \tag{6.8}$$

$$L_{ij} \leq S_i \text{ and } L_{ij} \leq S_j \ \forall Z_i, Z_j \in \mathbb{Z} \tag{6.9}$$

The constraint in Equation 6.4 states that the sum of the rates of flows between two switches should not exceed the link capacity. The constraint in equation 6.5 states that the number of flows entering and leaving switches which are neither destination nor sources of flow should be equal. The constraint in equation 6.6 assures a flow entering from source switch should reach the destination switch. The constraint in 6.7 asserts that a flow should not be assigned to a link that is connected to an inactive switch. Constraint in equation 6.8 models the relationship between the flows passing through a link and a switch. It asserts that if no flow is passing through all the links connected to switch Z_i, then change the binary variable $S_i = 0$. The constraint in

Equation 6.9 asserts that a link should be put in active state if and only if both of the switches it is connecting are active, otherwise, it is inactive.

6.4.3 MaxRESDN Heuristics Method

The energy efficient routing is an NP-hard problem [22, 25, 60]. Formal solutions can best fit for a small number of network switches but fail to scale up when the number is in the order of hundreds. However, in cloud data centers where the number of physical machines is in the order of thousand, the number of switches is in the order of hundreds. That triggers the need for heuristics algorithms that can run in reasonable polynomial time to provide sub-optimal solutions.

Algorithm 6 MaxRESDN ($\mathbb{G}$, $\mathbb{F}$,$\mathbb{U}$,$Umin$,$Umax$)

1: **Input:** Graph $\mathbb{G}$, set of traffic flow $\mathbb{F}$, utility of links $\mathbb{U}$, minimum utility $Umin$, and maximum utility $Umax$
2: **Output:** Modified utility of links $\mathbb{U}$ and graph $\mathbb{G}$
3: **for all** $f = (sr, ds, \lambda_f) \in F$ **do**
4: $path_f \leftarrow$ `PathMaxRESDN (sr,ds,`λ_f`)`
5: **for all** $e_{ij} \in path_f$ **do**
6: $U_{ij} \leftarrow U_{ij} + \frac{\lambda_f}{W_{ij}}$
7: **end for**
8: **end for**
9: **for all** $e_{ij} \in \mathbb{E}$ **do**
10: **if** $U_{ij} == 0$ **then**
11: $L_{ij} \leftarrow 0$
12: **end if**
13: **end for**

We propose the heuristics algorithm named MaxRESDN (Maximize RESDN). Algorithm 6 aims to maximize the RESDN value of the network as it assigns a path to each flow. The inputs for the MaxRESDN are the network represented as graph $\mathbb{G}$, the current utility of links $\mathbb{U}$, set of flows $\mathbb{F}$, the minimum utility $Umin$ and maximum utility $Umax$. Each flow $f = (sr, ds, \lambda_f) \in \mathbb{F}$ is expressed as source sr, destination ds, and flow rate λ_f which correspond to source address, destination address, and rate of f measured in bits per second (line 3). Line 6 increments the utilities of all the links

on the path assigned to flow f by the $\frac{\lambda_f}{W_{ij}}$ factor. Line 4 assigns each flow to the path that maximizes the RESDN among alternatives using the $PathMaxRESDN$ (sr,ds,λ_f) method. Lines 9 to 13 makes link e_{ij} inactive if its corresponding utility U_{ij} is 0.

Algorithm 7 PathMaxRESDN(sr,ds,λ_f)

```
1: Input: Graph 𝔾,utility of links 𝕌, minimum utility Umin, maximum utility
   Umax, source node sr, destination node ds, and flow rate λ_f
2: Output: path_f for flow f(sr, ds, λ_f) that maximizes RESDN
3: AllPath ← all paths between sr and ds
4: for all Path P ∈ AllPath do
5:     for all links e_ab ∈ P do
6:         if U_ab + λ_f > UMax then
7:             AllPath.remove(P)
8:         end if
9:     end for
10: end for
11: Path_f ← MaxRESDN(Allpath)
```

Algorithm 7 (PathMaxRESDN) returns the path that maximizes RESDN for a given flow f with source sr, destination ds, and flow rate λ_f. Line 3 initializes the *Allpath* list with all paths from sr to ds of flow f. Lines from 5 to 9, ensure the stability of the network by making the utilities of all links in each path plus the rate of flow of f not to exceed the $Umax$ value. While line 11 selects the path with maximum RESDN value and assign it to Path$_f$.

The complexity of the MaxRESDN algorithm depends on the total number of flows $|\mathbb{F}|$ and the PathMaxRESDN algorithm (7). Since the controller has a global view of the network, for each source destination switch pair in the network, all the paths from source to destination are stored in descending order with their RESDN value. The path with the maximum RESDN value then can be extracted in constant time. Therefore, the complexity of the MaxRESDN algorithm is O($|\mathbb{F}|$).

Chapter 7

RESDN: Experimental Analysis and Results

7.1 Experimental Platform

This section presents the experimental platform used to conduct our experiments. First, the network setup and the performance metrics are discussed, then utility based heuristics algorithms used for comparison are presented. Finally, the characteristics of the three OpenFlow protocol enabled switches we used in the experiments for measuring power consumption are presented.

7.1.1 Network Setup

The network setup is constructed using POX controller [90] and Mininet [14] network emulator installed on Ubuntu 16.04 64-bit. Mininet is used to create the topologies which comprise of hosts, switches, and links. Moreover, traffic is generated from each host by using the iperf command. The heuristics are implemented in the POX controller.

In POX, a switch is called datapath, where each datapath has a unique datapath ID (DPID). The first event of a switch is initiated when the switch establishes a connection channel, and this event is called ConnectionUP event. When the switch is

turned off or is disconnected from the network, it fires ConnectionDown event. Table 7.1 presents details of OpenFlow protocol event handling and the corresponding POX implementation.

Table 7.1: List of OpenFlow protocol events and the corresponding POX commands

Event	**POX command**	**Meaning**
ConnectionUp	-	Establishment of a new control channel with a switch
ConnectionDown	ofp_switch_features	Connection to switch is terminated
PortStatus	ofp_port_status	Controller receives port status message
FlowRemoved	ofp_flow_removed	Controller receives flow removed message
Statistics Events	ofp_stats_reply	Controller receives OpenFlow statistics reply
PacketIn	ofp_packet_in	packet at the port does not match
ErrorIn	ofp_error_msg	Controller receives OpenFlow error messages
BarrierIn	opft_barrier_reply	Switch finished processing sent by controller

Table 7.2 shows the list of messages that are sent between the controller and the switches, and their description. Whereas ofp_flow_mod and ofp_stats_request messages are generated by the controller, the ofp_packet_out messages are sent by the switches.

Table 7.2: Types of messages in POX controller

Message Sender	Command	Meaning
Switch	ofp_packet_out	Sending packets from the switch
Controller	ofp_flow_mod	Flow table modification
Controller	ofp_stats_request	Requesting statistics from switches

The types of statistics sent from the switch to the controller are listed in Table 7.3. These statistics range from the description of the switch, flow tables, to the status of queue in the switch.

7.1.2 Topologies, Traffic Traces, and Performance Metrics

Our experiments are conducted using real traces from SNDlib [91], in particular the GEANT dynamic traffic trace of the European research network.

Table 7.3: OpenFlow statistic types and the corresponding POX commands

Event	OpenFlow Stats Type
SwitchDescReceived	ofp_desc_stats
FlowStatsReceived	ofp_flow_stats
AggregateFlowStatsReceived	ofp_aggregate_stats_reply
TableStatsReceived	ofp_table_stats
PortStatsReceived	ofp_port_stats
QueueStatsReceived	ofp_queue_stats

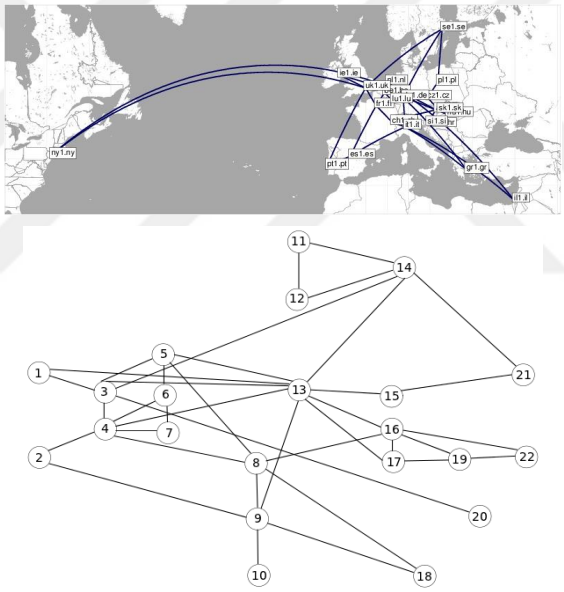


Figure 7.1: GEANT network topology a) Map view, b) Graph view

Figure 7.1 shows the GEANT topology used in our experiments [91]. The number of nodes and bidirectional links are 22 and 36, respectively. Traffic demand matrices are aggregated for 15 minutes from a 4-month duration trace. There exists total 11460 traffic demand matrices. The number of flows in each traffic demand matrix ranges from 82 to 462. To show traffic awareness in our experiments, different traffic volumes ranging from 20% to 90% with respect to the network capacity are used.

Table 7.4 presents the performance metrics of interest measured in our experiments, namely average power consumption of switches, RESDN value, percentage of

Table 7.4: Table of Metrics and Measurement Units

Metric	Unit / Description
Average Power Consumption of Switches	Watt
RESDN	metric as percentage
Links saved	percentage of links saved
Average path length	#hops
Throughput	Mbits/sec
Delay	milliseconds
Traffic Proportionality	[0,1]

links saved, average path length, throughput, delay, and traffic proportionality. The network performance is analyzed in terms of average path length, throughput and delay.

$$P_{switch} = P_{base} + P_{config} + P_{control} \tag{7.1}$$

Equation 7.1 shows the total power consumption of a switch (P_{switch}) as the sum of P_{base}, P_{config} and $P_{control}$ [92]. P_{base} is the power consumption for keeping the switch on without any active ports. The configuration power consumption P_{config} is calculated as

$$P_{config} = \sum_{i}^{N_{activePorts}} c_i.P_{port} \tag{7.2}$$

where c_i is the percentage of the maximum line speed of the port and P_{port} is the power consumption of a port at full capacity measured in watt.

Equation 7.3 shows the power consumption of control $P_{control}$ where $r_{PacketIn}$ and $E_{PacketIn}$ are the rate and energy consumption of $PacketIn$. $r_{FlowMod}$ and $E_{FlowMod}$ are the rate and energy consumption of the $FlowMod$ operations.

$$P_{control} = r_{PacketIn} x E_{PacketIn} + r_{FlowMod} x E_{FlowMod} \tag{7.3}$$

The ratio of energy saving metric for SDN is calculated as

$$RESDN = \frac{\sum_{\forall e_{ij}} X_{ij}}{\sum_{\forall e_{ij}} L_{ij}}$$

where L_{ij} is binary variable that denotes the status of edge e_{ij} and X_{ij} is a binary variable that tells if the utility of the link is between $Umin$ and $Umax$. Although, our primary goal is energy saving, RESDN also captures performance by the $Umax$ value.

The percentage of links saved is calculated as

$$\text{Links saved} = 100(1 - \frac{\sum_{\forall e_{ij}} L_{ij}}{|\mathbb{E}|}) \tag{7.4}$$

The average path length in terms of average number of hops is calculated as

$$\text{Average path length} = \frac{\sum_{\forall f} \sum_{\forall e_{ij}} (F_{ij})}{|\mathbb{F}|} \tag{7.5}$$

Throughput is calculated as the amount of data transferred per unit of time. In Mininet, we use Iperf command to measure the throughput of source and destination pairs. In the experiments, we measure the average throughput of all source and destination pairs.

Delay is calculated as the amount of time needed for data packets to be transferred from their source to destination. In our experiments, network delay is measured as the time it takes for the packets of a flow to start at the source and reach at the destination node. The delay is measured both on the Mininet side and POX controller.

Traffic proportionality is defined as the ratio of the traffic volume percentage to the network power consumption and is computed as

$$(\frac{\%\text{Traffic Volume}}{M\frac{\sum_{\forall Z_i} SP_i}{|\mathbb{Z}|} + N\frac{\sum_{\forall e_{ij}} L_{ij}}{|\mathbb{E}|}})(M+N) \tag{7.6}$$

where M:N is the ratio of the power consumption of the switches to the links. In these experiments, the ratio M:N is used as 3:1 [17].

7.1.3 Algorithms Used for Comparison

The heuristics algorithms we used to compare with MaxRESDN heuristics are listed in Table 7.5. The methodology in [53] formulates the problem with MIP and proposes four heuristic algorithms, namely Shortest Path First (SPF), Shortest Path Last (SPL), Smallest Demand First (SDF), and Highest Demand First (HDF). The first flow is assigned its corresponding shortest path, the succeeding flows are assigned paths where the change in energy consumption is minimized. Each heuristics algorithm uses a different criteria to sort the flows and processes them in the corresponding order. The four variations of the algorithms first sort the flows according to the shortest path first, shortest path last, smallest demand first, and highest demand first.

As explained in Section 6.3, the objective of NSP algorithm is to re-route flows passing through the under-utilized links to the next shortest path. NMU, on the other hand, chooses the path that has the link with maximum utility. While NSP gives priority to performance, NMU focuses on maximizing the utility of active links. Both NSP and NMU are not only ordering independent but can also be applied on top of other algorithm outputs to improve efficiency. The B heuristics is the application of NSP or NMU algorithms on top of the results of SPF, SPL, SDF, or HDF algorithms. It refers to the best outcome (B) in terms of energy efficiency.

Table 7.6 shows the values of $Umin$ and $Umax$ parameters used in the experiments for different percentage of traffic volumes. The values that maximize the number of

Table 7.5: Table of heuristics

Abbreviation	Description
SPF [53]	Shortest Path First
SPL [53]	Shortest Path Last
SDF [53]	Smallest Demand First
HDF [53]	Highest Demand First
NSP [17]	Next Shortest Path
NMU [17]	Next Maximum Utility
B [17]	Best Combination of NSP (NMU) with others
MaxRESDN	Maximize RESDN

links saved are determined by a greedy-based approach. A detailed analysis of these parameters is presented in Section 7.2.4.

Table 7.6: Parameters of $Umin$ and $Umax$ Used

Traffic volume percentage	20	30	40	50	60	70	80	90
$Umin$	31	28	30	25	20	19	15	12
$Umax$	82	85	90	90	92	95	95	95

7.1.4 Types of Switches

Table 7.7 shows the parameters used in the power calculation of two hardware switches (NEC PF 5240 and Zodiac FX) and one virtual switch Open vSwitch (OvS) which are OpenFlow enabled. NEC PF 5240 is a hybrid switch technology that adds OpenFlow protocol to the traditional network functionality [21, 92]. Open vSwitch (OvS) is a multilayer virtual switch designed for enabling network automation by supporting various protocols including but not limited to OpenFlow [92, 93]. Zodiac FX is an OpenFlow switch designed for small scale uses [94].

Table 7.7: Switch Power Consumption parameters

Parameters	NEC PF 5240 [21, 92]	OvS [92]	Zodiac FX [94]
Base[W]	118.33	48.7397	15
P_{port}[W]	0.52	Na	0.15
$E_{PacketIn}$ [μ W/packet]	711.30	775.53	775.53
$E_{FlowMod}$ [μ W/packet]	29.25	356.743	1455.13

7.2 Experimental Results

This section presents our extensive experimental results in terms of switch power consumption, RESDN and energy efficiency, network performance, and analysis of the link utility interval parameters.

7.2.1 Switch Power Consumption Results

Figure 7.2 shows the average power consumption of the switches for the NEC PF 5240 OpenFlow-based switch versus traffic volume. As compared to the best combination (B) of NSP or NMU with other algorithms (Table 7.5), MaxRESDN power consumption is on average 6 to 9 watts less than the B algorithm. As compared to NSP and NMU [17], MaxRESDN has shown up to 14.7 and 10.7 watts less in power consumption, respectively. This indicates that through maximizing the RESDN parameter, improvements in energy savings are achieved for all traffic volumes.

Figure 7.3 shows the power consumption of Zodiac FX switch for NSP, NMU, B, and MaxRESDN algorithms versus traffic volume. Similar to the previous experiments, the average power consumption of switches is proportional to the traffic volume. However, power consumption of MaxRESDN is on average 3.2, 2.3, and 2 watts less than NSP, NMU and B algorithms, respectively.

Figure 7.4 shows the power consumption of OvS switches versus traffic volume. Results show that as the traffic volume increases, the average power consumption of switches also increases. The MaxRESDN algorithm exhibits 5.8, 10, and 8.9 watts less energy consuming as compared to B, NSP, and NMU respectively.

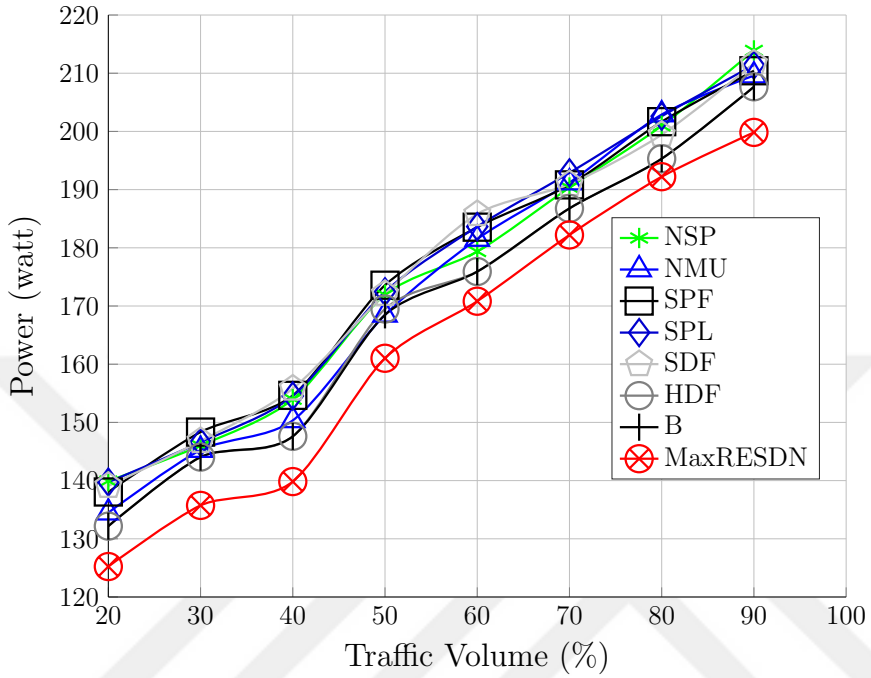


Figure 7.2: Power Profile for NEC PF 5240 switch in watt vs a range of traffic volume from 20% to 90%

Overall, Figures 7.2, 7.3, and 7.4 show that MaxRESDN is the most energy efficient as compared to the other algorithms (Table 7.5). Our findings indicate that maximizing the RESDN value helps minimizing the energy consumption. This is mainly because enforcing links to operate within the minimum and maximum utility parameters minimizes the number of active ports.

7.2.2 RESDN and Energy Efficiency Results

Figure 7.5 shows the RESDN values of all heuristic algorithms as the traffic volume increases. The MaxRESDN algorithm achieves the highest RESDN value which is 22% better for low traffic and 10% better than others for high traffic volume. This is because unlike the other algorithms whose objective is to minimize the number of links used in the network, MaxRESDN algorithm aims at maximizing the RESDN value, which increases the utility. In particular, for low traffic, it is more likely for flows that pass through overutilized links to be redirected to underutilized links.

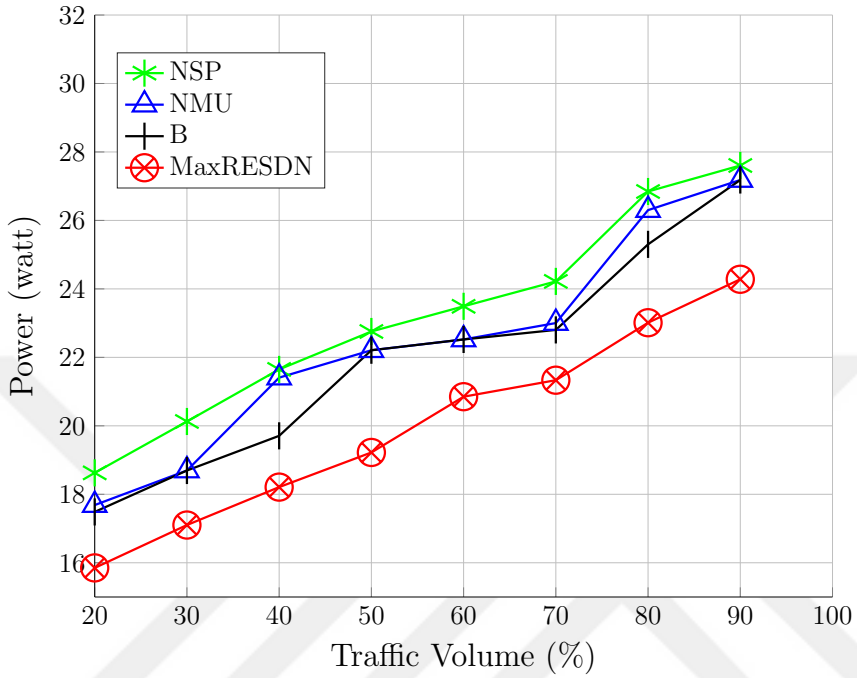


Figure 7.3: Power Profile for Zodiac FX switch in watt vs a range of traffic volume from 20% to 90%

Figure 7.6 shows the energy saving in terms of the percentage of links put to sleep. The B heuristics which is the best combination of the NSP or NMU algorithm exhibits the highest energy saving, while NSP and NMU have the lowest energy saving. MaxRESDN heuristics saves 38% for 20% traffic volume and 6.5% for 90% traffic volume.

Figure 7.7 presents the power consumption of all algorithms for the Zodiac switch. The results show that MaxRESDN switch power consumption is 16 to 22 watts which is on average 4 watts less than the SDF algorithm. MaxRESDN also demonstrated a power consumption on average 2 watts less than the B algorithm. These results show that by maximizing the RESDN value which is based on link utilities, by doing it frees ports on switches, hence it manages to reduce the power consumption of switches.

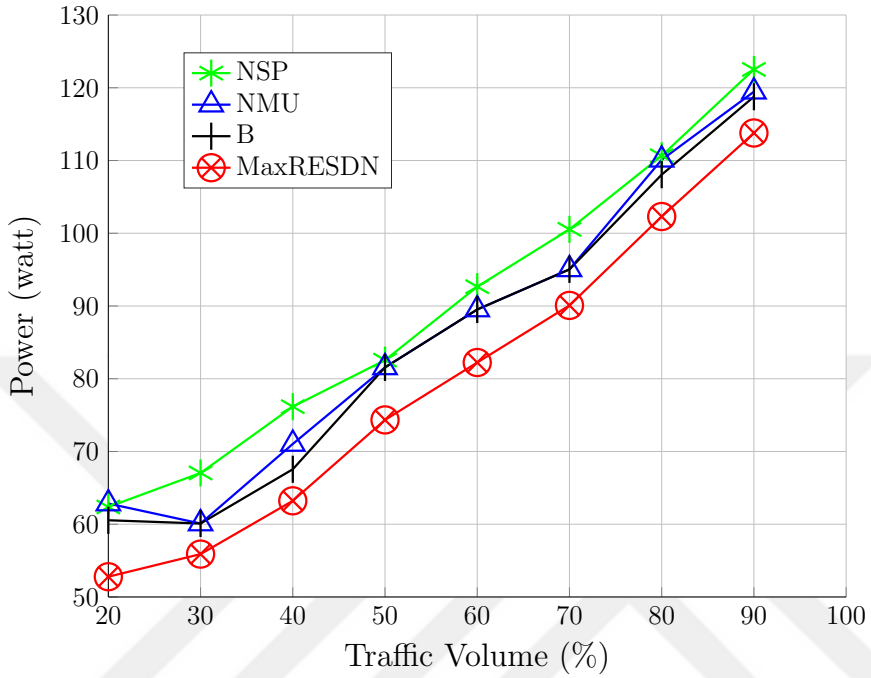
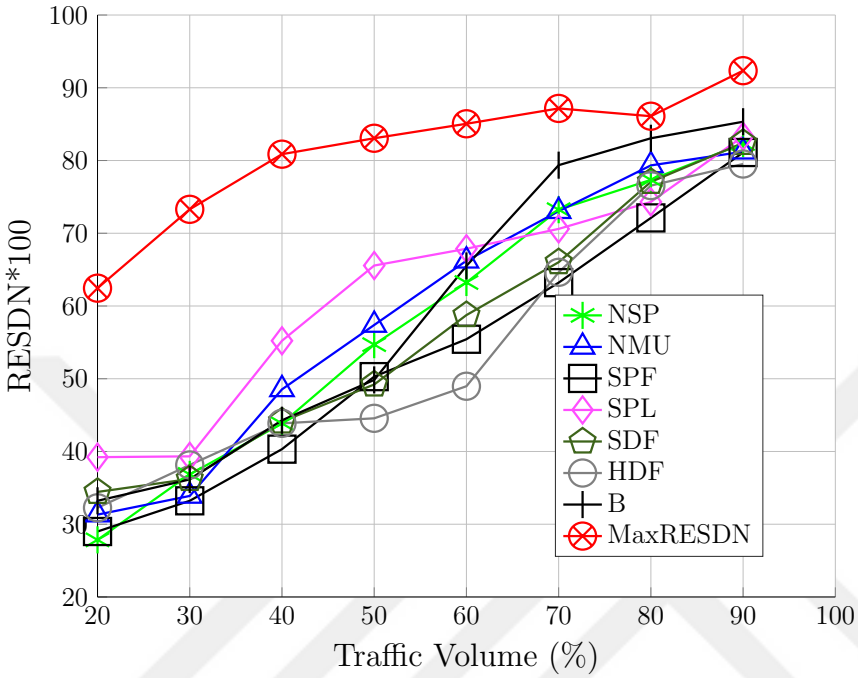


Figure 7.4: Power Profile for OvS switch in watt vs a range of traffic volume from 20% to 90%

7.2.3 Network Performance Results

Figure 7.8 shows the average path length (in terms of the number of hops) versus the traffic volume, and is calculated as given in equation 7.5. The NSP heuristics achieves the best path length since its objective is redirecting flows to the next alternative shortest path. The decisions are made periodically by selecting the underutilized links as candidates then redirecting the flows. B heuristics which is the combination of NSP and the best of SPF, SPL, SDF, and HDF in terms of energy saving shows improvement in the average path length. This indicates that our previous heuristics still can be applied on top of other algorithms and improve both performance and energy saving. The MaxRESDN heuristics performance is closer to NMU that has the objective of maximizing the utilities of the links, by directing the flows passing through underutilized links to more utilized links.

The experimental results show that NSP and NMU have the lowest average path length but least energy saving, since they first consider performance rather than



.tex .tex

Figure 7.5: RESDN Values NSP,NMU, SPF, SPL, SDF, HDF, B and MaxRESDN heuristic algorithms versus a traffic volume ranging from 20% to 90%

energy efficiency. The other algorithms in [53] exhibit better energy saving than NSP and NMU but worst average path length. This is because their objective is energy saving primarily. However, applying NSP and NMU on top of them increases energy saving. NSP and NMU balance the trade-off between performance and energy saving. The MaxRESDN heuristics, as compared to the other heuristics, has the maximum RESDN value. By maximizing a single RESDN value, the MaxRESDN has shown closer results to the best combination algorithm in terms of both energy saving and average path length. Assuming that the optimal value for *Umin* and *Umax* parameters is found for a given traffic, the MaxRESDN heuristics gives the maximum RESDN that achieves traffic proportional energy consumption and keeps the trade-off between energy efficiency and performance.

Figure 7.9 shows the throughput of the algorithms in Mbps with respect to the volume of traffic that ranges from 20% to 90%. The throughput measurement is done on Mininet using the iperf command. The average bandwidth of links is set

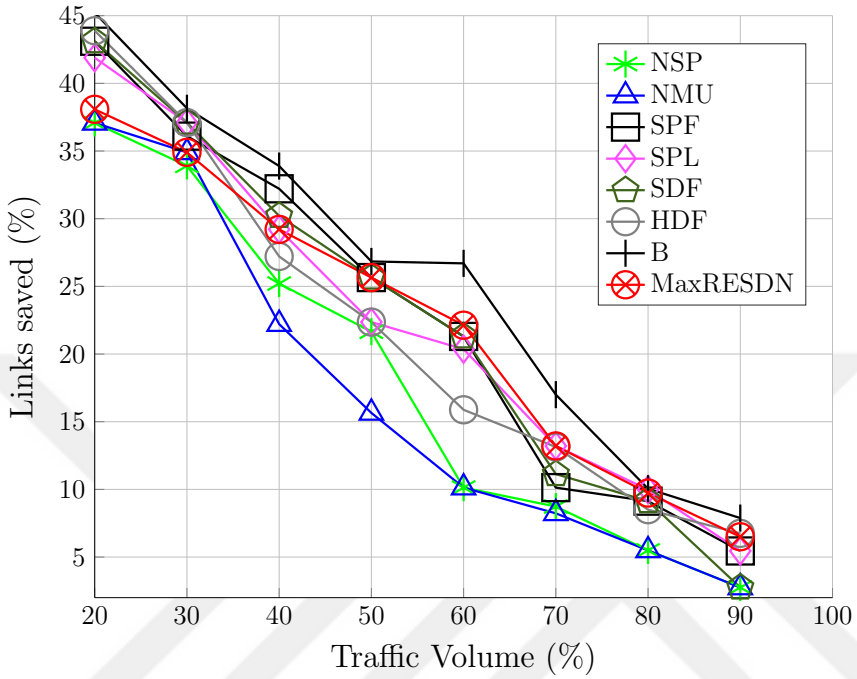


Figure 7.6: The percentage of links saved versus traffic volume ranging from 20% to 90%

to 100 Mbps and the average flow rate is 7.79 Mbps. Results show that the NSP algorithm exhibits the highest throughput for all the traffic range and is followed by NMU and MaxRESDN. This is because, for the three heuristics, the algorithms are initialized with the shortest paths. The maximum RESDN value by MaxRESDN as shown in figure 7.5 for all the traffic volume exhibits the least power consumption and also throughput close to that of NSP and NMU. For all the algorithms, the throughput decreases slightly as the traffic size increases after 30%. Under normal conditions where energy saving is not applied, throughput is meant to increase until the traffic volume is near 100% then decreases because of congestion. However, in energy saving routing algorithms, the attempt is to minimize the number of active links and switches. Therefore, the average throughput would decrease even when the traffic volume is near 50%. The trend of throughput in energy saving algorithms is different from non-energy saving performance focused heuristics.

Figure 7.10 shows the delay of the heuristics algorithms measured in milliseconds

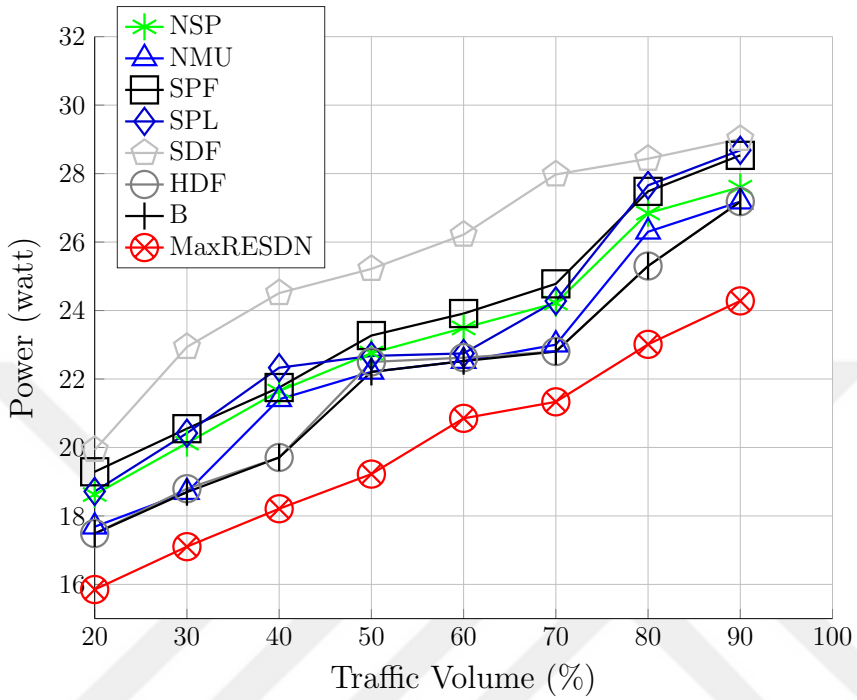


Figure 7.7: The average power consumption of the Zodiac switch measured in watts versus traffic volume ranging from 20% to 90%

with respect to traffic volume ranging from 20% to 90%. The NSP algorithm exhibits the least delay followed by NMU. The reason for this is because both NSP and NMU are initialized by the shortest paths. The SDF algorithm is 3 to 5 milliseconds worse in delay than the NSP algorithm. The MaxRESDN algorithm has a 1 millisecond to 3 milliseconds close to the NSP algorithm. It can be observed that for all of the energy saving heuristics, the delay tends to slightly increase with the increasing traffic volume. This is because all the energy saving algorithms attempt to minimize the number of links and switches used.

Figure 7.11 shows the traffic proportionality of heuristics for the range of traffic volume from 20% to 90%. The MaxRESDN heuristics have demonstrated the maximum traffic proportionality value. The B heuristics, which is an attempt of keeping the trade-off between performance and energy saving, has a closer traffic proportionality to our approach.

Figure 7.12 shows average traffic proportionality of all traffic volumes from 20%

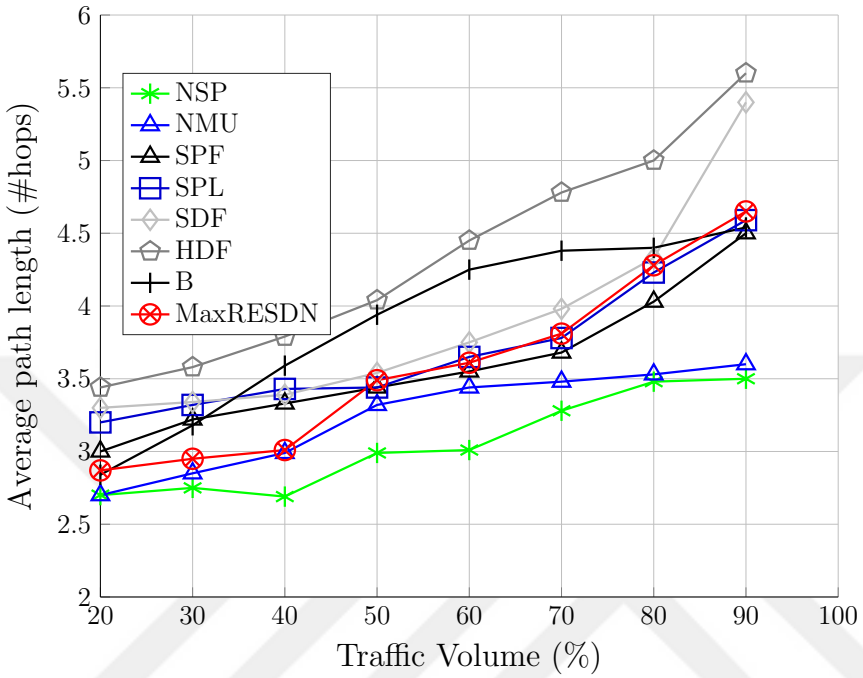


Figure 7.8: Average path length in terms of number of hops versus traffic volume ranging from 20% to 90%

to 90% for MaxRESDN algorithm compared with the other algorithms (in Table 7.5). The MaxRESDN heuristics exhibits the largest traffic proportionality in terms of link energy consumption. NSP algorithm has the lowest traffic proportionality. As compared to those heuristics which give priority to network performance such as NSP and SPF, our approach is 4 to 5% better in traffic proportionality, and 3 to 4% better in traffic proportionality than heuristics that give priority to energy saving and maximizing utility such as NSP and HDF.

7.2.4 Analysis of Utility Parameters

A challenge with the MaxRESDN algorithm is its dependence on the link utility interval parameters. The trends of link utility interval parameters $Umin$ and $Umax$ versus traffic volume and percentage of links saved are analyzed in this subsection. For the fixed $Umax$ value to 95%, we study the effect of the $Umin$ parameter on the percentage of links saved for traffic volume ranging from 20% to 90% $Umin$.

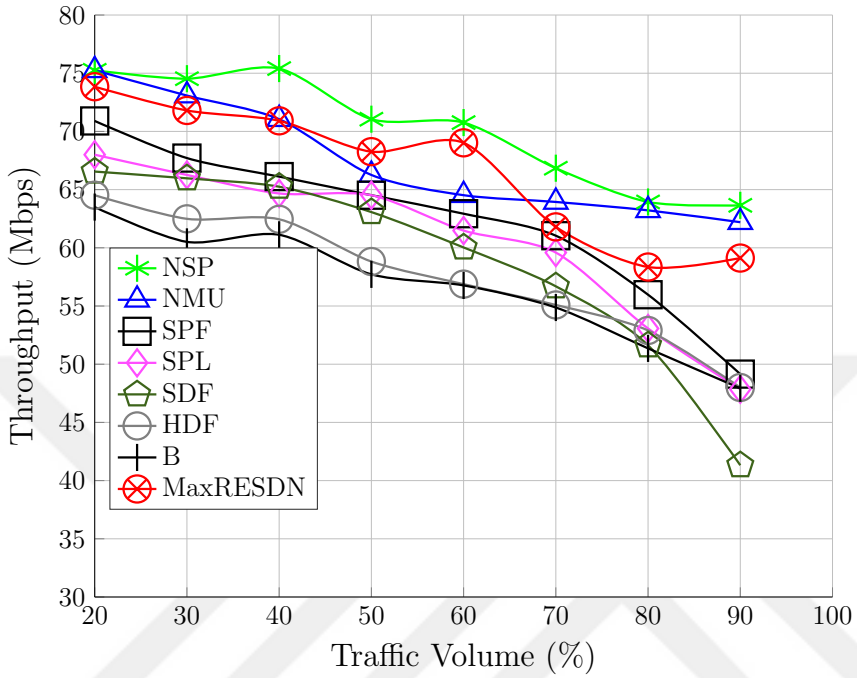


Figure 7.9: Throughput of heuristic algorithms in Mbps versus a range of traffic volume from 20% to 90%

Likewise, for the fixed $Umin$ value to 30%, we study the effect of $Umax$ value on the percentage of links saved. Understanding the relations and the trend for values of $Umin$ and $Umax$ versus the traffic volume would be significant in predicting the utility parameters for future use.

Figure 7.13 shows the effect of the $Umin$ parameter on the percentage of links saved for traffic volumes 20% to 90%. Each line represents a traffic volume. Fixing $Umax$ to 95%, and ranging $Umin$ value from 10% to 90%, the trend shows that the value of the $Umin$ that maximizes the links saved increases till it reaches a peak then drops. As the traffic volume increases the peak $Umin$ value increases. The analysis exhibits similar trend for different traffic volumes.

Figure 7.14 shows the effect of the $Umax$ value on the percentage of links saved for traffic volumes from 20% to 90%. Each line shows the traffic volume the experiment is conducted. The $Umin$ parameter is set to 30% and the $Umax$ value ranges from 40% to 100%. For every flow size, the $Umax$ value that maximizes the number of

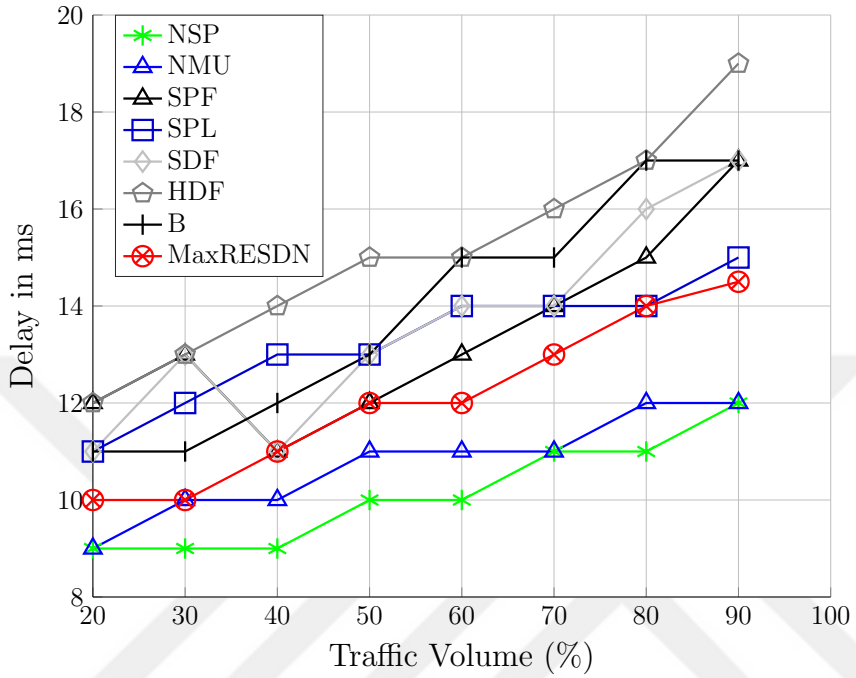


Figure 7.10: Delay of heuristic algorithms in milliseconds versus a range of traffic volume from 20% to 90%

links saved increases till it reaches its peak then it mainly remains constant except for the 20% traffic volume where there is a slight decrease.

Figure 7.15 shows the relationship between the traffic volume and the *Umin* value that maximizes the energy saving. It shows that as the traffic volume increases the *Umin* that maximizes energy saving increases. The reason for this is because the utilities of the links increase with traffic volume. For example when the percentage of traffic is 90%, the link with minimum utility is above 40%. Which means that if we pick *Umin* value less than 40%, the candidate list would be empty. Similarly as the traffic volume decrease to 20%, the link utilities starts from 12% and we will have more links in the candidate list. According to results on figures 7.13 and 7.15, the problem that need to be solved hence is to find or estimate the peak *Umin* value for a given traffic volume.

Figure 7.16 illustrates the trend of *Umax* value in relation to traffic volume by fixing the value of *Umin* the at 30%. It demonstrates that as the volume of traffic

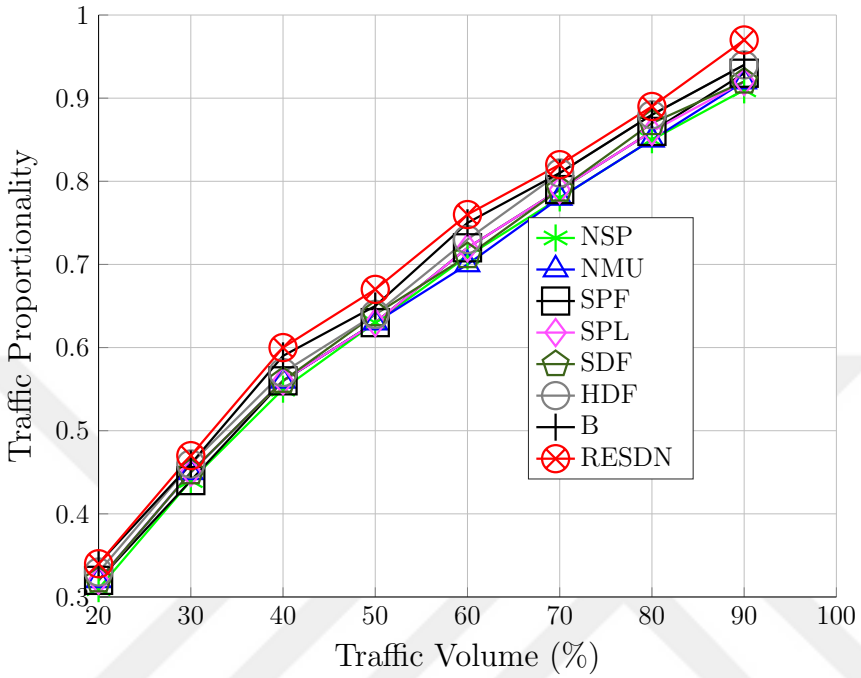


Figure 7.11: Traffic proportionality for the range of traffic volume from 20% to 90%

increases, the $Umax$ maximizing the energy saving exhibits an increasing trend. It also shows that for a traffic volume which is more than 72%, the $Umax$ value keep a constant value near 95%. According to the results depicted on Figure 7.16 and reffig:tanalysisofumin, picking the right value of $Umin$ and $Umax$ has a direct impact on the performance of the RESDN algorithm.

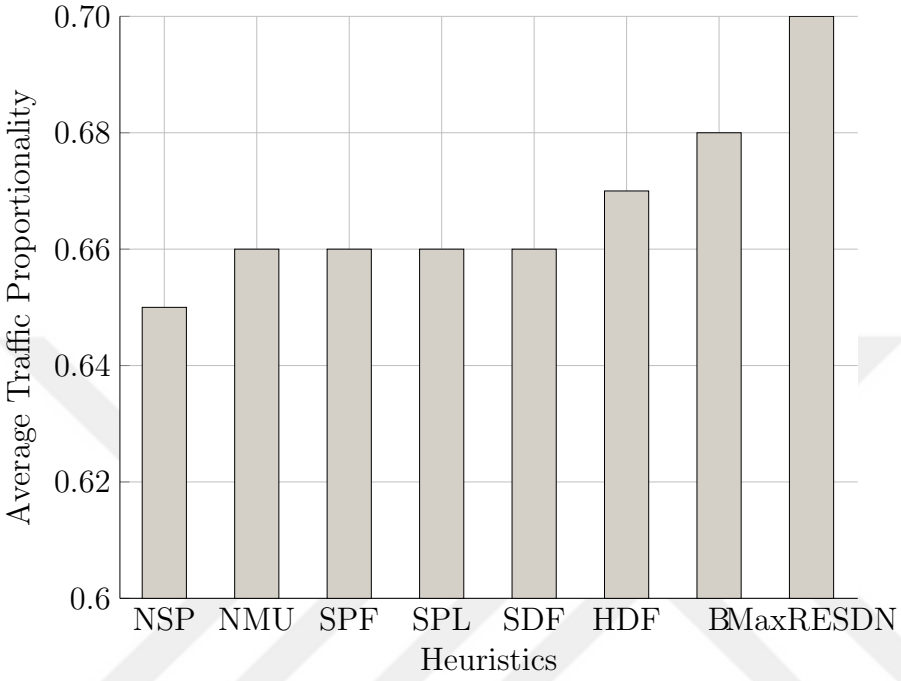


Figure 7.12: Average traffic proportionality of MaxRESDN as compared to other heuristics

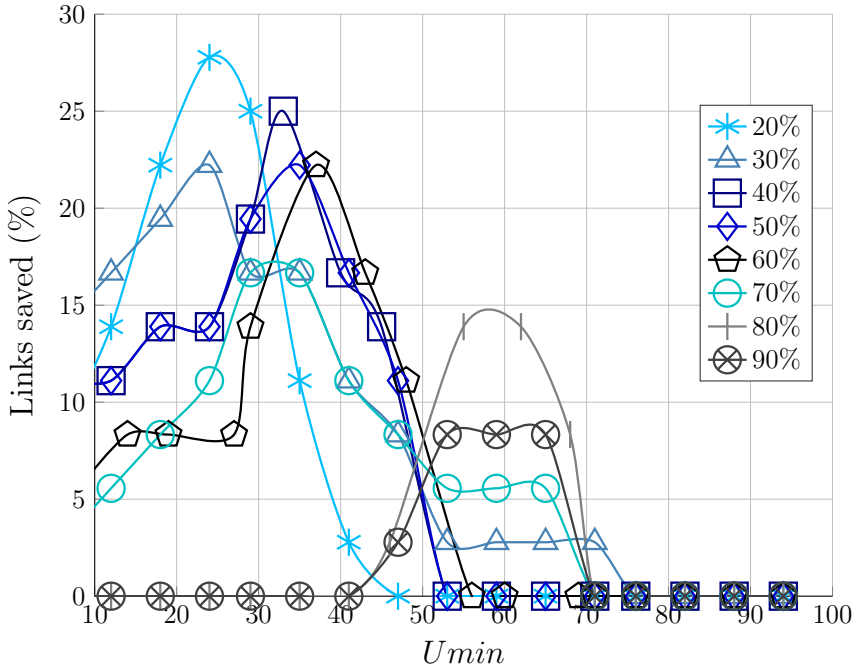


Figure 7.13: Effect of utility parameter $Umin$ values on the percentage of links saved for different traffic volumes ranging from 20% to 90% by fixing $Umax$ to 95%

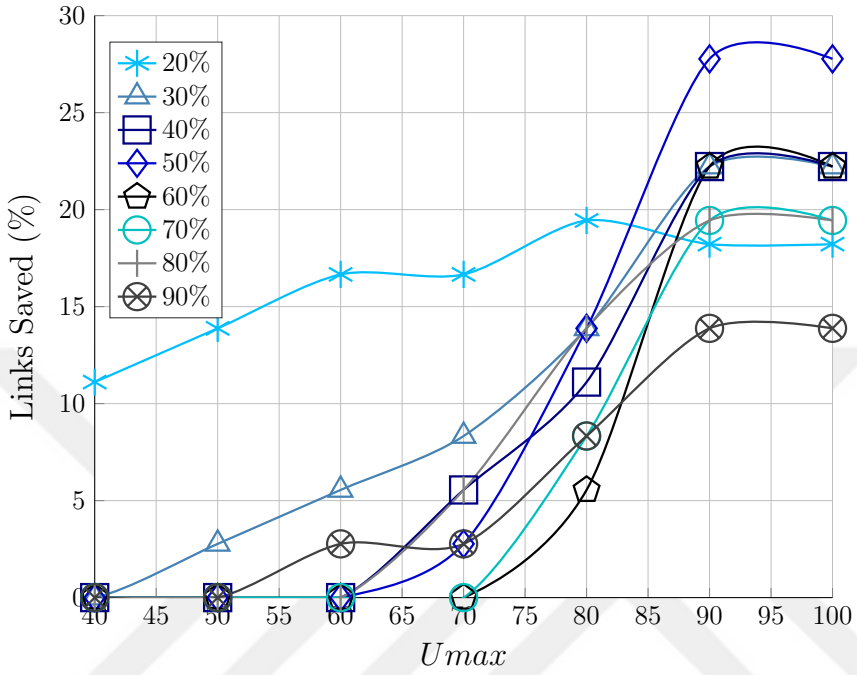


Figure 7.14: Effect of utility parameter $Umax$ values on the percentage of links saved for different traffic volumes ranging from 20% to 90% by fixing $Umin$ to 30%

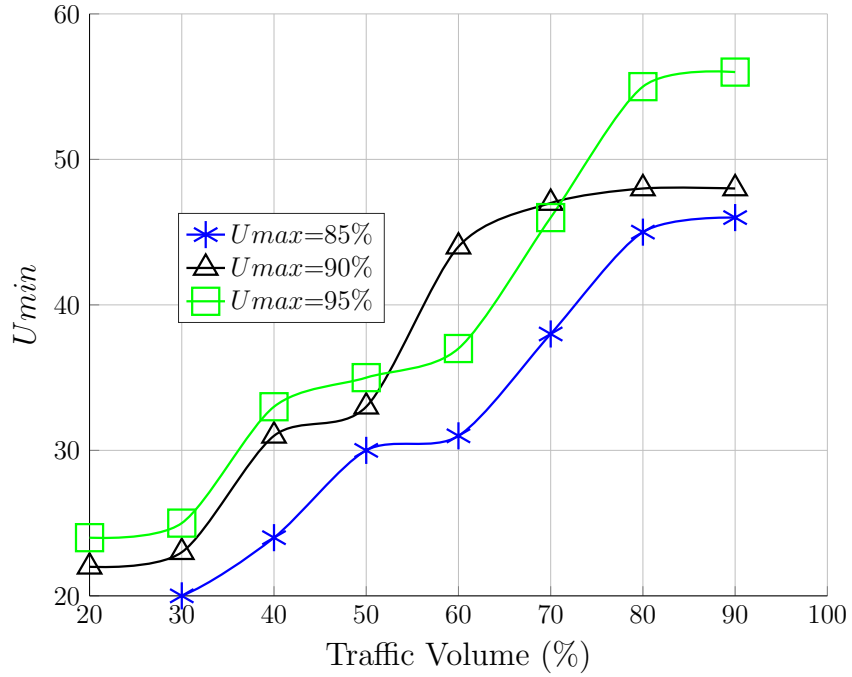


Figure 7.15: The effect of the traffic volume ranging from 20% to 90% on the value of $Umin$ and $Umax$ that maximizes the percentage of links saved fixing $Umax$ values to 85%, 90% and 95%

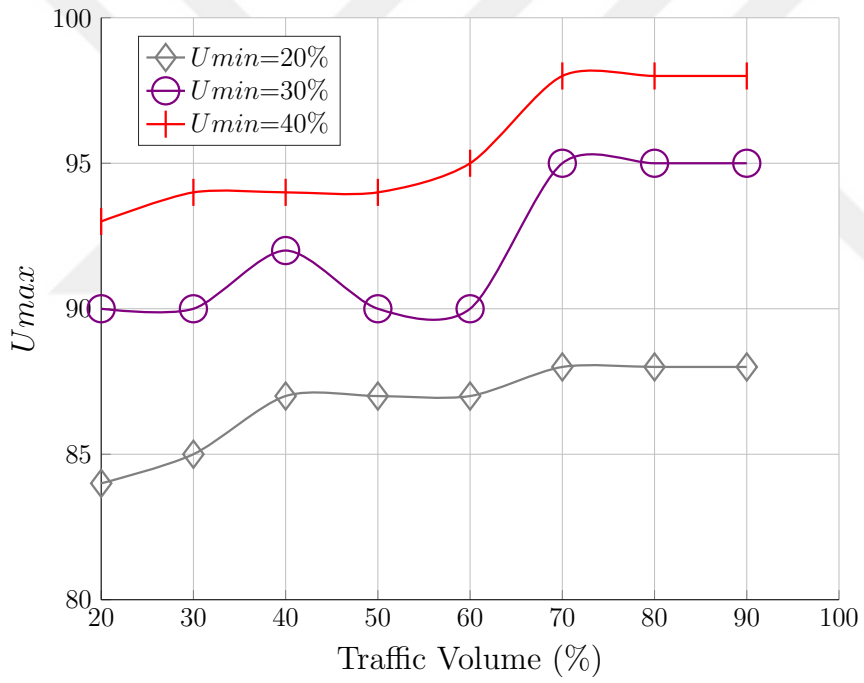


Figure 7.16: The effect of the traffic volume ranging from 20% to 90% on the value of $Umax$ that maximizes the percentage of links saved fixing $Umin$ values to 20%, 30%, and 40%

Chapter 8

HyMER: A Hybrid Machine Learning Framework for Energy Efficient Routing in SDN

8.1 Introduction

Several computing disciplines use machine learning as a tool to discover patterns in structured, semi-structured and unstructured data. Recently, there exists an increasing use of machine learning techniques in software defined networks (SDN) for traffic prediction, routing optimization, QoS prediction, resource management, and security [95–100]. Major ICT companies (e.g. Facebook, Yahoo, Microsoft, Huawei, Cisco, and Google) has adopted SDN paradigm in their data centers and network equipment designs [1, 2].

In SDN with the logically centralized controller, a massive amount of information is generated due to switch-controller communication. Events such as link up, link down, switch down, switch up, new packet arrival, trigger information to be sent to the controller. Likewise, flow modify, flow drop, and statistics related information are generated and can be stored within the controller. With the ever increasing amount of network information, machine learning techniques are formidable and play vital

role in discovering knowledge from the stored network information [97–99].

One of the most prominent challenges of the present world is energy since it has both economic and ecological issues. 10% of the global energy consumption is due to ICT sector out of which 2% is from network components. By 2020 the total electricity cost of cloud data centers is expected to increase by 63% [9, 101]. SDN enables to achieve traffic proportional energy consumption through dynamic re-routing of flows. The practical solution is to sleep/turn off underutilized components during low traffic load. However, there is a trade-off between network performance and energy efficiency since turning off network components for the sake of efficiency would have an adverse effect on network performance.

In this work, we propose a novel hybrid machine learning based framework named HyMER that combines the capabilities of SDN and machine learning for energy efficient routing. The three modules of HyMER framework are traffic manager, topology manager, and the learning machine. The framework combines the advantages of supervised and reinforcement learning models. Unlike other machine learning approaches that use a single algorithm and mainly focus on a single objective, HyMER, not only does exploit the advantages of supervised and reinforcement learning algorithms but also maintains the trade-off between network performance and energy efficiency. To the best of our knowledge, our approach is the first that proposes a hybrid machine learning solution for energy efficiency and network performance in SDN.

We have proposed integer programming (IP) formulations and heuristics to maintain the trade-off between network performance and energy efficiency in SDN [15–17]. However, the efficiency of the heuristics proposed, namely Next Shortest Path, Next Maximum Utility, and MaxRESDN, depends on the values of the link utility interval parameters that were previously determined by brute force. In this work, we use the supervised component of HyMER framework to predict the optimal values of the utility interval parameters to achieve the highest energy saving and acceptable network performance. The reinforcement component is used to achieve energy saving and

network performance by interacting with the network environment, hence, is able to handle the dynamic traffic and network status changes. The reinforcement component can learn from scratch or can be applied on the top of the supervised component.

The novelty of HyMER lies in the fact that it has the benefits of energy efficiency, network performance, dynamicity, and computational feasibility. Our approach not only maintains the trade-off between network performance and energy efficiency but also is able to capture the network dynamic nature in computationally feasible time. Switches, links and ports are energy saving capabilities in SDN. Energy efficiency is achieved by minimizing the power consumption of switches and the number of active links. Better network performance refers to the minimum average path length of flows, maximum throughput, and minimum delay. Dynamicity of an energy saving approach in SDN environment is the ease of adapting to the changing topology and traffic. Computational feasibility is the measure of how practical the proposed solution is in terms of time and space requirements.

The contributions of this work are as follows.

- We propose a hybrid three module machine learning framework, namely HyMER, for traffic proportional energy saving in SDN. The modules are Traffic Manager, Topology Manager, and Learning Machine. To the best of our knowledge, this is the first work to consider the learning machine as a module in an SDN controller for energy saving and network performance.

- Most of the machine learning approaches proposed for SDN are for traffic classification, routing, intrusion detection, or attack prediction. To the best of our knowledge, our HyMER framework is the first in applying machine learning to energy saving and network performance combined using both supervised and reinforcement learning.

- We propose a full-fledged supervised machine learning method that starts from feature extraction, applies feature reduction, and performs testing. Our results indicate more than 65% feature size reduction using Principal Component

Analysis (PCA). The supervised component predicts the link utility interval parameters with an accuracy of more than 70%. The proposed refine heuristics converges the predicted values to the optimal values with a speedup of 15X to 25X as compared to the brute force approach.

- We also propose a reinforcement learning method that minimizes energy consumption while keeping acceptable performance for dynamic routing in SDN. To the best of our knowledge, HyMER reinforcement component is the first to model both network performance and energy efficiency simultaneously. The reinforcement method converges to the maximum energy saving with a minimum of 100 to a maximum of 275 episodes. Episodes are the number of iterations which is the measure or time it takes for the reinforcement learning agent to reach the terminating state.

- We also demonstrate that combining the supervised and reinforcement methods not only does capture the dynamic change more efficiently but also increases the convergence speed. To the best of our knowledge, in the context of SDN, our approach is the first to combine supervised and reinforcement learning to jointly achieve energy saving and network performance. Initializing the reinforcement learning with the outputs of the supervised component speeds up the convergence by 2X on average.

- Experiments are conducted with Mininet and POX controller using real word network topologies and traffic traces from SNDLib [91]. In particular, Abiline, GEANT, and Nobel-Germany topologies and dynamic traffic traces are utilized. Switch power consumption is simulated using the SDN enabled switch NEC [21].

- HyMER heuristics has shown up to 50% link saving, and also exhibits up to 8.7 watts, 14.7 watts, and 10 watts less power consumption on average as compared to state-of-the-art utility based energy saving approaches for the Abiline, GEANT, and Nobel-Germany topology and real world traffic traces respectively.

- Our approach exhibits average path length, throughput, and delay closer to approaches which give priority to performance. However, it is on average 2 hops less in average path length, 15 Mbps more in throughput, and 5 ms less in delay as compared to approaches that give priority to energy saving. The comprehensive experiments demonstrate that HyMER maintains the trade-off between performance and energy efficiency.

The remainder of the chapter is organized as follows. Section 8.2 presents preliminaries about machine learning and related work. The HyMER framework is described in Section 8.3. The supervised and reinforcement learning components of the HyMER framework are presented in Sections 8.4 and 8.5 respectively.

8.2 Preliminaries and Related Work

8.2.1 Machine Learning Preliminaries

Table 8.1 shows the three categories of machine learning techniques namely, supervised, unsupervised, and reinforcement. In supervised learning, an example input data and its label (output) are provided. The goal of supervised learning is to infer the unknown function which maps the example training inputs into the output. The tasks in supervised learning are classification and prediction (regression) if the example output data type is categorical or numeric value respectively. In this approach, both example data and its corresponding output is provided. However, labeling the example data should be done by experts; hence, it is a time taking and labor-intensive task. In addition, such approaches have training scalability issues as the size of the training data gets larger.

Unsupervised learning, on the other hand, is only given the example unlabeled inputs but not their corresponding output. The task, therefore, is to uncover a hidden pattern in the unlabeled input data. Although such an approach avoids the labor-intensive and time taking labeling task, it may end up uncovering a pattern that

Table 8.1: Types of Machine Learning Techniques

Categories	**Task**	**Examples**	**Algorithms**
Supervised	Function from labeled data	Classification Prediction	Regression, SVM, Neural Net
Unsupervised	Pattern from unlabeled data	Clustering A ssociation	K-means, association rules
Reinforcement	Learn by interaction	Exploitation Exploration	Q-Learning, TD, DQN

may be of no interest. Reinforcement learning, on the other hand, does not have an example of input data. It learns by interacting with the environment through taking actions and collecting rewards. Such approaches are more suitable if the input data is delayed. However, reinforcement learning has a drawback of taking more time to converge [96].

In machine learning, feature extraction is a technique used to select a subset of data more relevant to finding interesting patterns. Feature extraction involves feature representation and feature reduction. The performance of machine learning method depends on the choice of features. Complex features require memory, computational power, and longer training time.

Feature reduction is a method of reducing the dimension of the feature set. Dimension reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables [102–104]. Major techniques used in machine learning are Principal Component Analysis (PCA) [105], Factor Analysis (FA), Projection Pursuit (PP), and Independent Component Analysis (ICA) [106]

8.2.2 Machine Learning in Networking

The use of machine learning techniques for energy efficiency in traditional networks has been studied [107], where the techniques are applied in assisting resource management, power distribution, demand forecasting, workload prediction, virtual machine placement prediction, memory assignment, CPU frequency, and traffic classification. The techniques range from supervised learning, unsupervised learning, reinforcement

learning, to hybrid methods.

There also exist several attempts in integrating machine learning to SDN. Approaches for enabling machine learning on SDN are discussed in [95, 108]. Table 8.2 shows the list of machine learning techniques and algorithms used in SDN, and their objectives. The objectives of using machine learning range from optimizing QoS [97] in terms of delay [99, 109, 110], congestion [111], and reliability [98] to ensure security in terms of attack prediction and classification. Supervised machine learning algorithms such as Neural Networks (NN), Hidden Markov Model (HMM), Bayesian networks are used. Reinforcement learning (RL) methods are also used to capture the dynamic nature of the network.

Table 8.2: Machine Learning Methods used in Software Defined Networking

Example	Algorithm	ML Type	Objective	Domain
[97]	NN	Supervised	QoS	General
[98]	HMM	Supervised	Reliability	Wi-Fi
[99]	NN	Supervised	Delay	General
[100]	RL based NN	RL	Loss rate and Security	Data center
[111]	QAR	RL	Congestion and fast delivery	Data center
[112]	NN and Bayesian	Supervised	DDoS Attack	General
[109]	RL	RL	Delay	General
[110]	DRL	RL	Delay	General

A meta-layered machine learning approach composed of multiple modules is proposed in [97], where the goal is to mimic the results of heuristics used in traffic engineering to maximize the quality of service (QoS). However, each neural network per module is trained separately and each trained model operates separately for each demand pair. The drawback of this approach is that it does not represent the relationships between the demands.

Seer is a configurable platform for network intelligence based on SDN, knowledge centric networking, and big data principles, where the goal is to accommodate the development of future algorithms and application that target network analytics [98]. It is also flexible in a sense that it allows high-level users to decide what network

information to use for their goals. By focusing on reliability, the platform aspires to provide a scalable, fault-tolerant and real-time platform, of production quality.

Another machine learning based approach NeuRoute is a dynamic framework which learns a routing algorithm and imitates its results using neural networks in real-time. NeuRoute is implemented on top of Google's TensorFlow machine learning framework and tested on POX controller. Experimental findings on GEANT topology show that the NeuRoute is faster than dynamic routing algorithms [99].

The work in [100] proposed a neural network based reinforcement learning method to design an SDN based secured overlay network for geographically distributed data centers communications over the public Internet. Experiment results show that the proposed Cognitive Routing Engine (CRE) finds sup-optimal QoS paths as compared to the optimal IP paths.

A reinforcement based QoS-aware Adaptive Routing (QAR) method for multi-tenancy controller environment is proposed in [111]. Considering a multi-layer hierarchical SDN, the approach leverages the scalability of the reinforcement learning approach in adapting to the changing environment. The objective of the approach is to maximize QoS in terms of congestion avoidance and fast packet forwarding. The reward function for the reinforcement learning is based on QoS.

Machine learning in SDN is also used in predicting the host to be attacked [112] using C4.5 decision tree classifier, Bayesian Network, Decision Table, and Naive-Bayes algorithms. Prediction of DDoS attack using neural network is implemented in NOX controller [113].

In SDN, the controller is logically centralized. Practically, centralized control is achieved through multiple controllers working in a distributed but coordinated manner. Reinforcement learning is used to aid the dynamic routing in SDN via distributed controllers. Emulation results show that the proposed algorithm not only captures a dynamic traffic demand but also exhibits improved QoS in terms of loss rate and delay as compared to OSPF (Open Shortest Path First) routing protocol implemented on SDN controller [109].

The work in [110] uses Deep Reinforcement Learning (DRL) algorithm to optimize routing by minimizing network delay. Given the traffic matrix, the approach uses deep learning as a black-box to find all optimal paths from all sources to all destinations. The DRL attempts to learn the paths that minimize network delay iteratively. Experimental results show that the approach performs better than traditional dynamic routing algorithms with respect to computation time.

In contrast to the existing machine learning based solutions proposed for SDN which are mainly on security, traffic classification [95], and QoS in terms of delay, congestion, and throughput, our framework models performance and energy efficiency at the same time using both supervised and reinforcement learning. We present a method of representing network traffic as features, perform feature size reduction using mathematically proven techniques, provide heuristics to increase the accuracy of the prediction to 100%. Moreover, we have also modeled a dynamic energy efficient routing algorithm for SDN using reinforcement learning. In our approach, the link utility interval parameters are predicted for the MaxRESDN heuristics algorithm [16].

8.3 HyMER Framework Description

We propose HyMER framework that utilizes machine learning techniques to achieve traffic proportional energy efficiency in SDN. Our approach is hybrid in a sense that it combines the best features of supervised and reinforcement learning. The objectives of the HyMER hybrid framework are to jointly formulate energy efficiency and network performance, to propose generalized heuristics algorithms, and to apply machine learning approaches on SDN controller that learn from traffic, topology and solution history.

Figure 8.1 illustrates the HyMER framework consisting of three modules: traffic manager, learning machine and topology manager. The information of the traffic generated by the applications is passed to the traffic manager that stores details of traffic information in terms of source-destination pairs, rate, starting time of each traffic

flow, and the total amount of flows in the repository. The status of the network and the topology information are stored in the repository. The learning machine should give an optimal sub-graph based on the traffic volume by learning from historical data in case of the supervised, or learns by interacting with the network environment in case of the reinforcement learning. Low traffic load would result in a sub-graph with a smaller number of active links and switches as compared to a sub-graph in the case of high traffic load. The topology manager module is responsible for retrieving information about the organization and status of the network components and store it in the repository. It also keeps track of cost information of links and forwarding switches. If a network component fails or is out of service, the topology manager updates the global topology information.

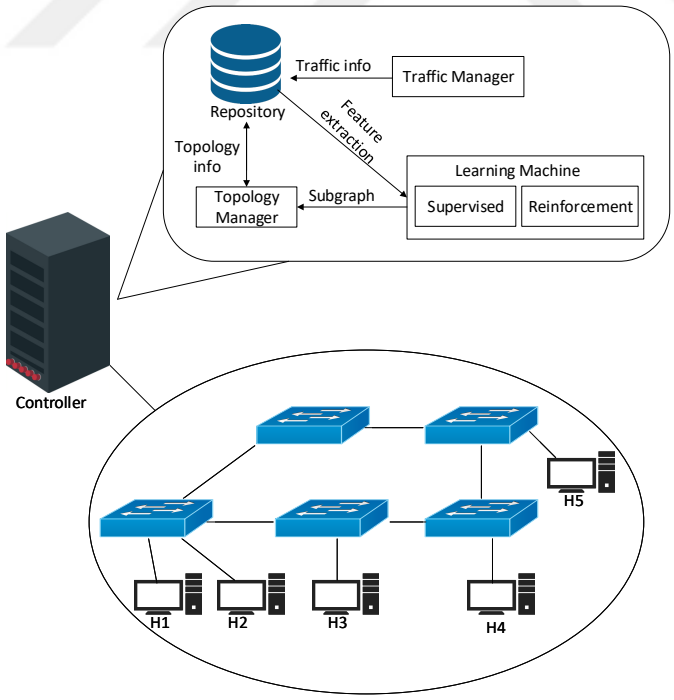


Figure 8.1: HyMER: A Hybrid Machine Learning Framework for Energy Efficient Routing in Software Defined Networking

There exist two machine learning components inside the learning machine module, namely supervised learning component and reinforcement learning component. The supervised learning component learns from historical labeled training data. This com-

ponent is appropriate when there exist large amount of input and their corresponding output data. Deciding on the amount of training data is not a straightforward task but collecting as much data as possible is of practical significance. The more data we have the better the learning is. If there is small amount of training data, the model may have highest training accuracy but would perform less when tested with unseen new data. This is called overfitting. The reinforcement learning module works even if there is no sample training input and output data. It learns by iteratively interacting with the network environment by taking actions and accumulating rewards.

8.4 The Supervised Learning Component of the HyMER Framework

Figure 8.2 illustrates the supervised learning component of the HyMER framework. This component consists of three stages: feature extraction, training, and testing. The feature extraction stage extracts features from the traffic, topology, switch, and link data, and represents them using a matrix to perform size reduction. In this work, we represent the traffic matrix (X) where each row i is a traffic snapshot (Sn_i) and the columns are source destination switch pairs. Each value in the matrix represents the traffic flow rate between the source and destination for the corresponding traffic snapshot. The training stage sets the hyperparameters of the training model using cross-validation, and then builds a training model. The testing stage makes a prediction on the next sub-optimal graph that is proportional to the traffic volume. The refining prediction part of the testing stage improves the predicted state using a heuristics algorithm.

8.4.1 Feature Extraction

The first stage of the supervised learning component has two parts; feature representation and feature reduction. The type of data that we use and how we model it for the training stage is called feature representation. Network modeling and traffic

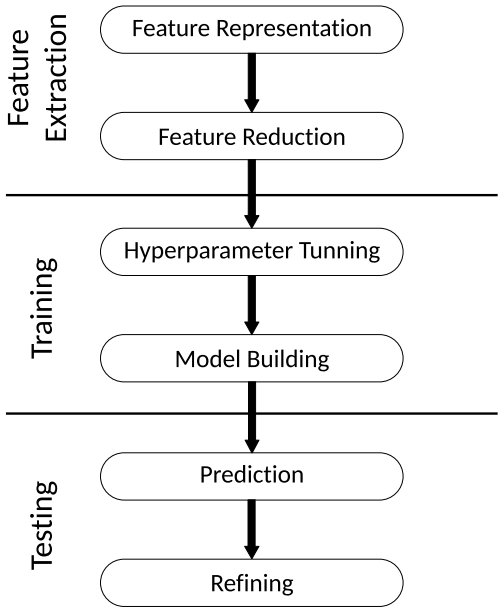


Figure 8.2: The supervised component of the HyMER framework

representations need to reflect the network environment. Feature representation has a direct effect on the performance of the machine learning model. Feature reduction deals with minimizing the size of the data without losing valuable information.

Feature Representation

The network is represented as a graph where the nodes and the edges correspond to the switches and the links, respectively. A traffic flow is represented by the source node, destination node, and the flow rate. If the number of switches in the network is m, then the column size of the traffic matrix X is represented as

$$d = mx(m-1)$$

If we extract n number of snapshots for training data, then the dimension of the traffic matrix X

$$|X| = nxd$$

where n is the total number of traffic demand snapshots taken periodically.

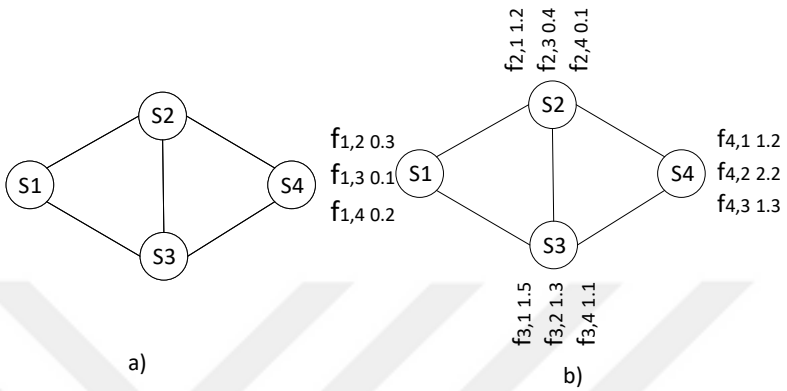


Figure 8.3: Example of network topology and traffic snapshot a) The network is represented as a graph where the switches are the nodes and the edges are the links b) A traffic snapshot

Figure 8.3.a shows an example graph representation of a network topology with 4 switches and 4 links. Figure 8.3.b represents both the network topology as a graph and traffic demand from each node and their corresponding flow rates. Switch $S1$ has flow demands $f_{1,2}$, $f_{1,3}$, and $f_{1,4}$ with flow rates 0.3, 0.1, and 0.2 units. Likewise, there exist three distinct traffic flows from $S2$, $S3$, and $S4$.

Table 8.3: Traffic feature extraction and representation of snapshots of a network

Snapshot	$f_{1,2}$	$f_{1,3}$	$f_{1,4}$	$f_{2,1}$	$f_{2,3}$	$f_{2,4}$	$f_{3,1}$	$f_{3,2}$	$f_{3,4}$	$f_{4,1}$	$f_{4,2}$	$f_{4,3}$
Sn_1	**0.3**	**0.1**	**0.2**	**1.2**	**0.4**	**0.1**	**1.5**	**1.3**	**1.1**	**1.2**	**2.2**	**1.3**
Sn_2	1.5	1.9	1.1	0.9	1	2.5	2.9	2.3	1.1	2.9	1.2	1.8
Sn_3	1	1.2	2.7	1.4	3	2	2.5	1.4	1.9	3	1.3	1.2
Sn_4	1	1.3	2.8	1.8	2.1	2.2	2.1	1.3	2.2	1.6	2.8	0.8

Table 8.3 shows how the traffic feature is extracted and represented in our approach for the example topology in Figure 8.3.a. Each row in the table shows a snapshot of the traffic. Each column represents the source destination pairs and the value represents traffic flow rates between the source and destination. A snapshot of the traffic and network status is taken periodically. For example, $f_{1,2}$ column represents the traffic flow with switch $S1$ as the source switch and $S2$ as the destination switch.

The first row, snapshot Sn_1 is the representation of the network snapshot of Figure 8.3.b. The flow rate $f_{1,2}$ between switches $S1$ and $S2$ is 0.3. Similarly, the flow rates of $f_{1,3}$, $f_{1,4}$, $f_{2,1}$, and $f_{2,3}$ are 0.1, 0.2, 1.2, and 0.4 units. Likewise, Sn_2, Sn_3, and Sn_4 are example traffic snapshots of the network topology presented in Figure 8.3.

Feature Size Reduction

We utilize PCA (Principal Component Analysis) for feature size reduction in the supervised learning component of HyMER. PCA is a linear combination of optimally-weighted observed variables. The outputs of PCA are these principal components whose numbers are less than or equal to the size of the original feature space. The principal components are orthogonal to each other. PCA is commonly used in face recognition, image classification, and unsupervised predictions.

Algorithm 8 shows the steps used in PCA for feature size reduction [114]. The inputs to the algorithm are the traffic matrix X^{nxd} as stipulated in Figure 8.3 and Table 8.3, and the number of principal components k. Line 1 computes the mean vector $\bar{X}$ of X. The dimension of $\bar{X}$ is equal to the feature dimension d. Line 2 mean normalizes the traffic matrix. Mean normalization is necessary because it makes each feature component have the same standard deviation which helps all principal components to have equal weight. The next step of PCA is to compute the covariance matrix of the mean normalized data and compute the eigenvectors V and eigenvalues E as stipulated on lines 3 and 4. Line 5 orders the V based on eigenvalues E in descending order. The eigenvector corresponding to the maximum eigenvalue is at the first position while the eigenvector corresponding to the minimum eigenvalue is at the end of the list.

The next step in the PCA algorithm is to prepare the projection matrix W with the top k principal components. However, selecting the value of k is a significant step in PCA and a challenging task. Small k value reduces the feature size significantly but preserves fewer information of the original data. The variance of the principal components shows the direction of the eigenvector corresponding the to maximum

Algorithm 8 PCA: Reduce the traffic matrix X^{nxd} to X^{nxk} and produce the projection matrix W^{dxk}

Input: Traffic matrix X^{nxd} and k the number of principal components
Output: Feature data X^{nxk} and projection matrix W^{dxk} where k is the number of principal components and d is

1: $\bar{X} \leftarrow \sum_{i=1}^{d} X_i$ ▷ mean of X
2: $T \leftarrow$X - $\bar{X}$ ▷ mean normalize T
3: $C \leftarrow \frac{1}{n}(X - \bar{X})^T(X - \bar{X})$ ▷ C is the covariance matrix
4: $V, E \leftarrow eig(C)$ ▷ computer eigen value and vector
5: $V \leftarrow sort_{desc}(V, E)$ ▷ sort V based on E
6: $W \leftarrow eigenvecs^k$ ▷ Projection matrix W^{dxk}
7: $X^{nxk} \leftarrow XW$ ▷ Project X on W space

eigenvalue that carries most of the information in the original unreduced data. The larger the variance the more information the principal components carry. If we use the whole principal components the variance would be closer to 100%, and if the number of principal components chosen does not carry any information about the whole matrix, the value becomes 0. The variance decreases while moving from the first (largest) component to the last one. Line 6 computes the projection matrix W^{dxk}. Line 7 reduces the dxn dimensional matrix X to dxk by projecting it over the eigenspace W. The outputs of the algorithm are the projected matrix X^{nxk} and the projection matrix W^{dxk}. The projection matrix W^{dxk} would also be used in the testing stage of the supervised learning component to transform the unseen new traffic matrix X^{txd}_{new} to X^{txk}_{new}.

8.4.2 Training

The training stage of HyMER has two parts: hyperparameter tuning and model training. We use Principal Component Analysis (PCA) to reduce feature dimension and train a regression model. The hyperparameter needed for the PCA algorithm is the number of principal components k and the hyperparameters for the regression model are the learning rate $\alpha \in (0, 1]$ and number of iterations. A learning rate close to 0 takes longer training time. Since regression is an iterative approach, the

number of iterations is the maximum number of repetitions the algorithm performs before it converges or terminates. We use n fold cross-validation technique to pick the right parameter k. In n-fold cross-validation [115], where our case is 10-fold cross-validation, the original sample is randomly partitioned into ten equal size sub samples where nine of the ten sub-samples are used for training and the remaining one is used as validation. This process is repeated by making each sub-sample as a validation set only once and as a training set nine times. The ten results from the ten-folds are then averaged to produce a single estimation. The 10-fold cross validation is done for various k values until the optimal k value that gives the maximum accuracy is found. After tuning the parameters, the next step is to build a regression model by using all the training dataset. The regression model is then ready to be used for predicting the out puts of unseen data.

8.4.3 Testing

Testing refers to applying the trained model to predict the values for unknown new traffic matrix. There exist two steps in this stage; prediction and refining. For a new traffic matrix Y^{txd} with t number of snapshots, we use the projection matrix W^{dxk} to reduce its size to Y^{txk} that is computed as follows.

$$Y^{txk} = Y^{txd}W^{dxk}$$

Then, we use the regression model we built (as described in Subsection 8.4.2) for prediction. In order to increase the accuracy of the model, Algorithm 9 is utilized. The rationale behind the Refine algorithm is to increase the predicted $Umin$ by step size parameter α until the energy saving decreases and to decrease the value of the predicted $Umax$ by α until the energy saving remains constant. The steps taken by increasing $Umin$ and decreasing $Umax$ by α until we get the optimal value is based on the analysis of $Umin$ and $Umax$ parameters of the MaxRESDN heuristics in [15].

The inputs to the algorithm 9 are predicted $Umin_0$, predicted $Umax_0$, step size α, and threshold β. The threshold parameter β is the terminating condition for the algorithm that measures the energy saving difference between previous and current $Umin$ and $Umax$ values.

Lines 2, 3, and 4 calculate the efficiency of the energy saving algorithm MaxRESDN for the $Umin$ parameter value of $Umin_0$, $Umin_0-\alpha$, and $Umin_0+\alpha$ respectively. The MaxRESDN heuristics maximizes the Ratio for Energy Saving (RESDN) value of the network environment. The larger the RESDN value the larger the energy saving and the better the network performance is [15]. Lines from 5 to 9 if the change in $Umin$ changes the energy saving. Line 11 sets the terminating condition for refining the value of the predicted $Umin$ by checking if the difference between the energy saving using the current $Umin$ and the previous $Umin$ is not greater than the threshold β. Lines from 13 to 15 alliteratively reduce the $Umax_0$ value until the energy saving does not change according to line 17. The optimal values of $Umin$ and $Umax$ are found on lines 16, and 17.

Algorithm 9 Refine: Improves the predicted parameters $Umin_0$ and $Umax_0$ values for better efficiency

Input: Predicted parameters $Umin_0$ and $Umax_0$, change α, threshold β.
Output: Improved parameters $Umin$,$Umax$

repeat
  $EE_{curr} \leftarrow EE(Umin_0, Umax_0)$
  $EE_{prev} \leftarrow EE(Umin_0 - \alpha, Umax_0)$
  $EE_{next} \leftarrow EE(Umin_0 + \alpha, Umax_0)$
  if $EE_{prev} < EE_{next}$ **then**
    $Umin_0 \leftarrow Umin_0 + \alpha$
  else
    $Umin_0 \leftarrow Umin_0 - \alpha$
  end if
  $EE_{new} \leftarrow EE(Umin_0, Umax_0)$
until $ABS(EE_{curr} - EE_{new}) \leq \beta$
while $EE_{curr} \geq EE(Umin_0, Umax_0 - \alpha)$ **do**
  $Umax_0 \leftarrow Umax_0 - \alpha$
  $EE_{curr} \leftarrow EE(Umin_0, Umax_0)$
end while
$Umin \leftarrow Umin_0$
$Umax \leftarrow Umax_0$

8.5 The Reinforcement Learning Component of HyMER Framework

8.5.1 Reinforcement Learning Preliminaries

Reinforcement learning is a machine learning technique where an agent learns about the states of an environment through iterative interaction by taking a set of actions. Each action the agent takes transforms the environment from one state to another. The environment gives a reward value to each action taken by the agent. The agent learns the actions that maximize the cumulative reward values [96].

Figure 8.4 shows how the reinforcement learning algorithm works. The agent interacts with the environment by taking actions. The action taken changes the environment from one state to the other. The agent then observes the state change in the environment, and a reward value is given to the action. The goal of the

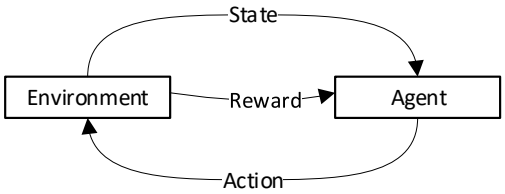


Figure 8.4: Reinforcement learning

agent is to maximize the cumulative reward value through taking actions attractively, observation of the environmental changes, and updating the reward for each action.

Despite a longer convergence time in comparison to supervised approach, reinforcement learning is the most appropriate machine learning method to model dynamic routing as compared to supervised and unsupervised learning. Unlike the supervised learning technique (described in Section 8.3) which depends on historical labeled training data, reinforcement-learning does not need labeled training data but it rather learns from scratch by interacting with the environment.

A prominent and difficult step in modeling any problem with reinforcement learning is the definition of action, states, and a reward function. The size of the set of actions and the number of states has a direct impact on the performance of the algorithm. The reward function not only defines the objective to be achieved but also plays a major role in the convergence of the algorithm.

8.5.2 Reinforcement Learning Model for SDN

Figure 8.5 shows the way we modeled the reinforcement learning for energy efficient dynamic routing in SDN. The controller is an agent that learns by interacting with the environment by taking actions and getting rewards. The environment comprises the switches, links, and the traffic. Traffic is defined as a set of flows each with source address, destination address, and flow rate in bits/sec. The state of the environment at a given time is defined as the status of the links (active/inactive), the utility of links, the status of the switches, and the traffic demand. The action is the list of links (i.e. route) that the agent takes to deliver traffic flow from the source to destination.

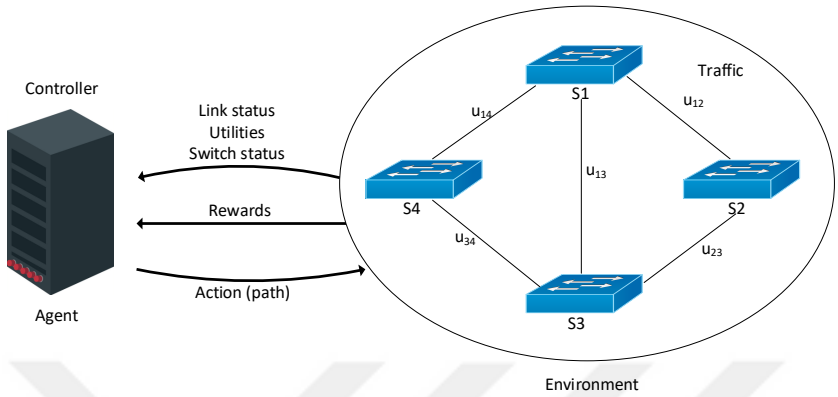


Figure 8.5: SDN controller reinforcement model implementation for energy efficient routing

The controller interacts with the environment to learn the most energy efficient path for a flow with source and destination. Each action A_t at time t changes state S_t to S_{t+1}, where t is an episode in the learning. Rewards are calculated for each action based on how energy efficient and how close the links to be chosen are to the destination node. We model the reward at episode t as a matrix R_t where each row represents traffic in terms of source-destination pair and each column represents the set of actions which are the set of links. The reward matrix R_0 is initialized as follows.

$$r^0_{fx} \in \mathrm{R}_0 \leftarrow \begin{cases} MaxR, & f_{dst} \text{ directly connected} \\ & \text{link } x \\ -MaxR, & x \text{ is not active} \\ 1, & Umin \leq U(x) \leq Umax \\ U(x) - Umin, & U(x) < Umin \\ Umax - U(x), & U(x) > Umax \end{cases} \tag{8.1}$$

where f represents a source-destination pair of traffic demand, f_{dst} represents the destination of flow f, x represents a link, and U(x) represents the utility of link x. Equation 8.1 stipulates that the maximum reward value $MaxR$ is given for the link x which is directly linked to the destination of flow f. If the utility of the link x is

less than the $Umin$ or greater than $Umax$, it would be penalized with U(x)-$Umin$ and $Umax$-U(x) respectively. A maximum punishment of $MaxR$ is given if x is not active. For links with utilities between $Umin$ and $Umax$, a reward value of 1 is given.

The goal of the reinforcement learning is to maximize the expected value of the cumulative rewards and is calculated as

$$E(\sum_{e=0}^{\infty} \gamma^e R_e) \tag{8.2}$$

where R_e is a reward given in episode e, $\gamma^e \in [0, 1]$ is discount rate at the episode, and E is the expected cumulative reward value. An important trade-off that needs to be balanced in reinforcement learning is the issue of exploration and exploitation. Exploration mainly focuses on discovering new solutions to the problem at the cost of taking the risk of getting less rewards. In our case, the agent can choose an alternative link with minimum reward value aiming to find or explore a better reward in the future.

Exploitation on the other hand takes a risk averse decision which tends to keep the already acquired rewards and only takes alternative solution if it is better than the previous one. In our routing case, the agent would choose another link only if it can increase the cumulative reward value. Exploitation is more of a greedy approach.

The discount parameter γ^e is the way we maintain the trade-off between exploration and exploitation at episode e. The discount parameter decreases the reward at episode e by some factor between 0 and 1. The larger the discount parameter, the lesser the reward is discounted, the closer the discount parameter is to 0, the more the discount on the reward is. Hence, by setting the discount parameter to values less than one, the agent can pick a link with lower reward value over higher ones. There exist two approaches to set the value of the discount rate. One approach is to set $\forall\gamma^e \leftarrow \Gamma$, the other one is to have different value for each episode. In the latter, it is first set at high value and then reduced at later steps like simulated annealing. In

this work, we set the value of $\gamma^e \leftarrow \Gamma$.

8.5.3 Formulation of Energy Efficient Dynamic Routing with Q-learning Algorithm

Q-learning is a model-free value based reinforcement learning algorithm [116]. Q learning defines a Q-value for each state-action pair, and iteratively maximizes the Q-values by interacting with the environment.The state in our case is the status the network components, and utility of links. State-action pair is the state of the network and the next link to be chosen or the flow among the alternatives by considering one state-action ahead. The Q-value of the next state-action is a measure of how valuable the next action would be given the current state.

Algorithm 10 Q-Routing: Uses Q-learning algorithm to solve dynamic energy efficient routing algorithm

Input: Q-table $\leftarrow R_0$,Discount rate γ, Learning rate α,and State S
Output: Updated Q, Episode E

1: **repeat**(e$\in$ E) ▷ For each episode
2: Choose Action A
3: Observe r^e, $S^{'}$ for taking action A
4: Q(S,A) $\leftarrow$ Q(S,A)+ α [r^e+ γ $\max_{A'}$ Q($S^{'}$,$A^{'}$)-Q(S,A)]
5: $S \leftarrow S^{'}$
6: Update R ▷
7: **until** S is terminal

Algorithm 10, Q-Routing, shows a modification of the Q-learning algorithm for dynamic routing in [117, 118] and [119] which were designed for finding the shortest path and minimizing packet delivery time respectively. Our Q-routing algorithm is different from them in two ways. First, the reward function is used to give the highest value for the shortest path, but in our case, it gives the highest value if the path keeps the utility of the links between the minimum and maximum utility values. This keeps the trade-off between energy efficiency and performance. Second, the actions in our case are the links, but in the [118, 119], the nodes/switches are the actions.

The inputs to the Q-Routing algorithm are the initial Q-table which is equivalent to the reward matrix R_0. Line 2 chooses an action randomly to start, then observes the reward for the randomly picked action A, and also observe the next state S'. This step is repeated until the flow reaches it destination. Through the iterative process, paths of a flow my change depending on the cumulative reward.

The complexity of the Q-routing algorithm 10, given the agent knows the topology and all actions, is $O(|A||S|^2)$ [120]. In this context, the actions are the number of links and the states are the traffic demand pairs multiplied by the state of each link. If u and v are the number of links and the number of traffic demand pairs in the network respectively, the number of states equals uv, hence, the complexity of the Q-routing algorithm is $O(u^3v^2)$.

8.6 Usage of HyMER Framework Algorithms

There exist three algorithms in the HyMER framework. PCA and Refine algorithms are parts of the supervised component of the framework. Q-routing algorithm is part of the reinforcement learning component of HyMER. Except for PCA, the Refine and Q-routing algorithms are heuristics. Figure 8.6 shows the usage of these algorithms and we refer them as HyMER heuristics or HyMER in short.

As illustrated in Figure 8.6, the choice to start with the supervised component or the reinforcement component depends on the availability of historical data. If there exist historical data, the supervised component applies feature representation, feature dimension reduction with PCA, model training, testing, and the refine heuristics is used to predict the optimal $Umin$ and $Umax$ parameters. The parameters are then used by the MaxRESDN algorithm. If there is a change in the environment due to new flow arrival or update in the network status, instead of doing predictions and running the MaxRESDN algorithm again, the reinforcement component acts relatively and responds easily to the changing environment. In the case where there is no historical data, the reinforcement component, using the Q-routing heuristics can learn energy

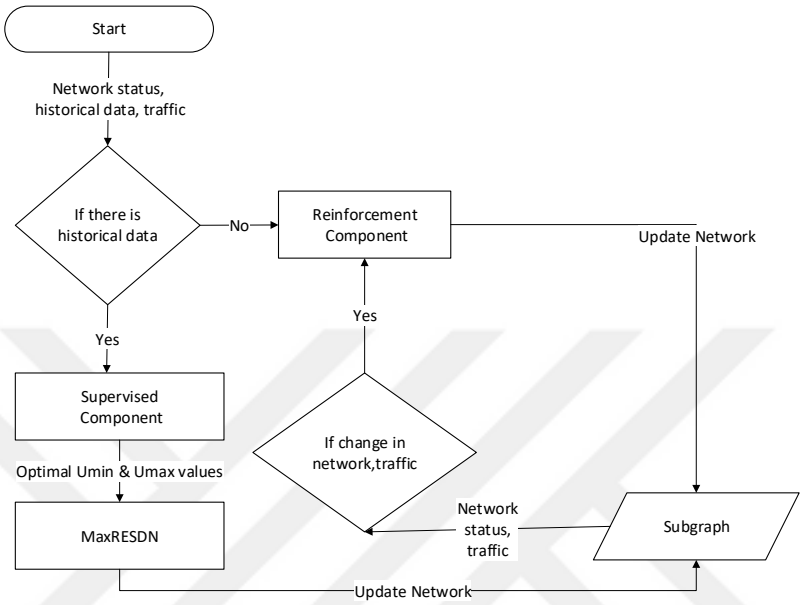


Figure 8.6: The usage of HyMER framework algorithms

efficient paths for traffic flows.

Chapter 9

HyMER: Experimental Analysis and Results

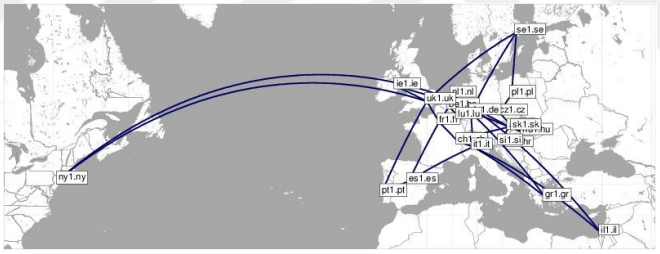
This section presents the comprehensive experiments we conducted to evaluate the HyMER framework. First, we present our experimental platform, the performance metrics of interest, and the real world topology and traffic traces used in the experiments. Then, we discuss briefly the energy efficiency approaches used for comparison. The next three subsections present results of the supervised component, reinforcement component, and network energy efficiency and performance, respectively. In total there exist ten metrics out of which four of them (i.e., accuracy, variance, feature size reduction, and speed up) are for the supervised component, one (i.e., Maximum Q value) is for the reinforcement component, two (i.e., links saving, and switch power consumption) for energy and power saving, and three (i.e., average path length, throughput, and delay) for network performance.

9.1 Experimental Platform, Metrics, and Datasets

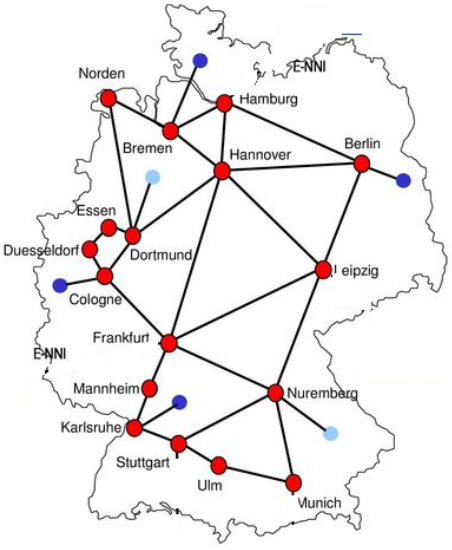
The experimental platform is based on POX controller and Mininet [14] network emulator installed on Ubuntu 16.04 64-bit. The topologies are created on Mininet, and the heuristics are implemented on POX controller. Our experiments are conducted

using real world traces from SNDlib [91], in particular, the Abilene, GEANT, and Nobel-Germany network topologies and dynamic traffic traces.

(a)



(b)



(c)

Figure 9.1: Images of the network topologies used in the experiments a) Abilene, b) GEANT, and c) Nobel-Germany

Figure 9.1 shows the images of the network topologies used in the experiments. Abilene is a network that connects cities in the United States of America. GEANT is a European network that connects countries in the EU including England. Nobel-Germany is a network among the cities in Germany.

Table 9.1: Topologies and Traces

Topology	Nodes	Edges	Avg Deg.	Feature Size	Snapshot Minutes
Abilene	12	15	2.5	132	5
GEANT	22	36	4.35	462	15
Nobel-Germany	17	26	3.06	272	5

Table 9.1 presents the topologies, traffic characteristics and the size of the datasets used in our experiments. The features are extracted from snapshots of the network aggregated in 5, 15 and 5 minutes for the Abiline, GEANT, and Nobel-Germany dynamic traffic traces. The features are represented as a matrix where each row is an m x (m-1) vector representing the rates between a source and destination pairs. For the Abilene topology with 12 nodes, the dimension of the feature is 12 x 11 = 132. Accordingly, the feature sizes for GEANT and Nobel-Germany are 462 and 272 respectively. We train the models for traffic volumes ranging from 10% to 90%. The percentage of traffic volume is calculated with respect to the total bandwidth capacity of the links. For these experiments, the average bandwidth of links is set to 100 Mbps and the average flow rate is 11.36 Mbps, 7.79 Mbps, and 9.56 Mbps for Abiline, GEANT, and Nobel-Germany topologies and traffic traces, respectively.

Table 9.2 presents the performance metrics of interest that are evaluated in the experiments. They consist of the accuracy of the predictor, feature size reduction due to PCA, cross-fold validation to pick the optimal number of principal components, speedup of the predictor and the refine algorithm as compared to the brute force method, energy efficiency, and average path length. Accuracy is calculated as

Table 9.2: The Performance Metrics

Metric	Meaning	In terms of
Accuracy	The accuracy of the predictor	%
Feature Size Reduction	Percentage of feature size reduced by using PCA	%
Variance	The percentage of information retained in the reduced feature size	%
Speedup	How fast the Refine algorithm works as compared to brute-force	X
Max-Q Value	The maximum Q-value achieved per each iteration	scalar
Power	Average power consumption of switches	Watt
Average path length	Average length of a path for source destination demand pair	#hops
Links saved	Percentage of links saved	%
Throughput	Average throughput of flows	Mbits/sec
Delay	Average Delay	milliseconds

$$100 * (1 - \frac{|TV - PV|}{TV}) \pm \epsilon \tag{9.1}$$

where TV is the true value of the parameter, PV the predicted value of the parameter, and ϵ is the error tolerated. In the experiments, we set the value of ϵ to 3%. Speedup is calculated as $100/N$ where N is the number of times the energy saving algorithm (MaxRESDN) is run before the Refine algorithm gets the optimal value. The power consumption of a switch is calculated as the sum of the power for the base, configuration and control [92].

$$P_{switch} = P_{base} + P_{config} + P_{control} \tag{9.2}$$

Equation 9.2 shows the total power consumption of a switch (P_{switch}) as the sum of P_{base}, P_{config} and $P_{control}$ [92]. P_{base} is the power consumption for keeping the switch on without any active ports. The configuration power consumption P_{config} is calculated as

$$P_{config} = \sum_{i}^{N_{activePorts}} c_i.P_{port} \tag{9.3}$$

where c_i is the percentage of the maximum line speed of the port and P_{port} is the power consumption of a port at full capacity measured in watt.

Equation 9.4 shows the power consumption of control $P_{control}$ where $r_{PacketIn}$ and $E_{PacketIn}$ are the rate and energy consumption of $PacketIn$. $r_{FlowMod}$ and $E_{FlowMod}$ are the rate and energy consumption of the $FlowMod$ operations.

$$P_{control} = r_{PacketIn} x E_{PacketIn} + r_{FlowMod} x E_{FlowMod} \tag{9.4}$$

Table 9.3 shows the power consumption parameters of the NEC PF250 switches that are used in the experiments.

Table 9.3: Switch power consumption parameters

Parameters	Values
Base[W]	118.33
P_{port}[W]	0.52
$E_{PacketIn}$ [μ W/packet]	711.30
$E_{FlowMod}$ [μ W/packet]	29.25

The percentage of links saved is calculated as

$$\text{Links saved} = 100(1 - \frac{\sum_{\forall e_{ij}} L_{ij}}{|\mathbb{E}|}) \tag{9.5}$$

The average path length in terms of average number of hops is calculated as

$$\text{Average path length} = \frac{\sum_{\forall f} \sum_{\forall e_{ij}} (F_{ij})}{|\mathbb{F}|} \tag{9.6}$$

Throughput is calculated as the amount of data transferred per unit of time. In Mininet, we use Iperf command to measure the throughput of source and destination pairs. In the experiments, we measure the average throughput of all source and destination pairs.

Delay is calculated as the amount of time needed for data packets to be transferred

from their source to destination. In our experiments, network delay is measured as the time it takes for the packets of a flow to start at the source and reach at the destination node. The delay is measured both on the Mininet side and POX controller.

9.2 Algorithms Used for Comparison

The heuristics algorithms used to compare MaxRESDN heuristics are listed in Table 9.4. The methodology in [53] starts by formulating the problem with MIP, then proposes four heuristic algorithms. The first flow is assigned its corresponding shortest path, the succeeding flows are assigned paths where the change in energy consumption is minimized. Each heuristics algorithm uses a different criteria to sort the flows and processes them in the corresponding order. The four variations of the algorithms first sort the flows according to the Shortest Path First (SPF), Shortest Path Last (SPL), Smallest Demand First (SDF), and Highest Demand First (HDF).

Table 9.4: Table of Heuristics

Abbreviation	Description
SPF [53]	Shortest Path First
SPL [53]	Shortest Path Last
SDF [53]	Smallest Demand First
HDF [53]	Highest Demand First
NSP [17]	Next Shortest Path
NMU [17]	Next Maximum Utility
B [17]	Best Combination of NSP (NMU) with others
HyMER	ML assisted MaxRESDN [15]

The objective of NSP algorithm is to re-route flows passing through the under-utilized links to the next shortest path. NMU, on the other hand, chooses the path that has the link with maximum utility. Whereas NSP gives priority to performance, NMU focuses on maximizing the utility of active links. Both NSP and NMU are not only ordering independent but can also be applied on top of other algorithm outputs

to improve efficiency. The B heuristics is the application of NSP or NMU algorithms on top of the results of SPF, SPL, SDF, or HDF algorithms. It is the best in terms of energy efficiency.

HyMER is the machine learning assisted version of MaxRESDN heuristics which achieves energy saving, network performance, and traffic proportionality by maximizing the link utility interval metric named Ratio for Energy Saving in SDN (RESDN). Both the supervised and the reinforcement components of the HyMER achieve results consistent with MaxRESDN.

9.3 Results of the Supervised Learning Component of HyMER

Figure 9.2 show size reduction, cross-validation accuracy and variance of the model for the Abilene, GEANT, and Nobel-Germany topology and traces. Size reduction is inversely proportional to accuracy and variance. The percentage of principal components we picked from the 10-fold cross validation are 30%, 32% and 35% which correspond to accuracy values 78%, 79%, 80% and feature size reduction of 70%, 68%, 65% for the Abilene, GEANT and Nobel-Germany topologies and traces, respectively. The value of k chosen for the PCA is 40, 148, and 96 for the three topologies and traces. The number of principal components is up to 5% larger than where the accuracy and the size reduction plots intersect. The choice is carefully made so that the model would not over-fit the data and at the same time contain at least 80% of the information in the original data.

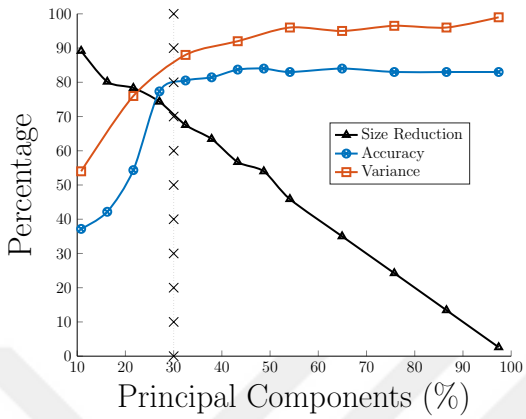
Figure 9.3 shows the accuracy of predicting $Umin$, $Umax$, $Umin/Umax$ ($Umax$ given $Umin$ is known), $Umax/Umin$ ($Umin$ given $Umax$ is known) versus for the GEANT data set. The accuracy of predicting $Umin$ ranges between 68% to 75%. For the Abilene trace, $Umax$ prediction accuracy is 3 to 5% better than $Umin$'s prediction. An interesting observation from this experiment is that the accuracy of $Umin$ and $Umax$ increases if $Umax$ and $Umin$ are known apriori. In case of the

GEANT topology trace, a prior knowledge of $Umin$ increases the prediction accuracy of $Umax$ by at least 15%. The accuracy of the predictor is independent of the traffic volume.

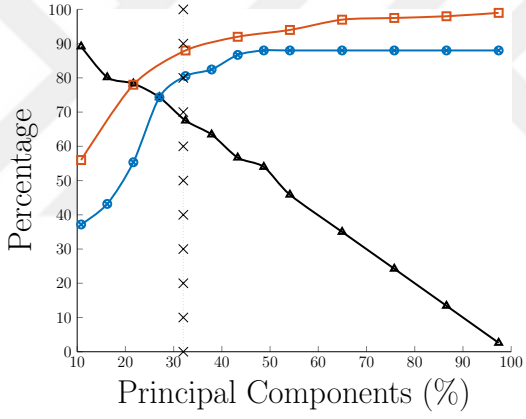
Table 9.5: Refine algorithm speedup for convergence of the $Umin$ and $Umax$ parameters of MaxRESDN heuristics algorithm as compared to the brute force method

Traffic	Abilene		GEANT		Nobel-Germany	
(%)	$Umin$	$Umax$	$Umin$	$Umax$	$Umin$	$Umax$
10	17.86	21.43	18.37	15.43	14.29	18.37
20	16.07	22.69	16.77	16.77	18.37	16.77
30	14.84	20.3	18.37	16.77	16.77	16.07
40	16.77	19.29	20.3	19.29	21.43	19.29
50	17.53	19.29	16.07	20.3	19.29	19.29
60	16.07	19.29	16.77	16.77	18.37	21.43
70	15.43	25.71	18.37	24.11	14.84	18.37
80	16.07	20.3	20.3	22.69	14.29	16.07
90	14.29	20.3	21.43	24.11	16.77	25.71

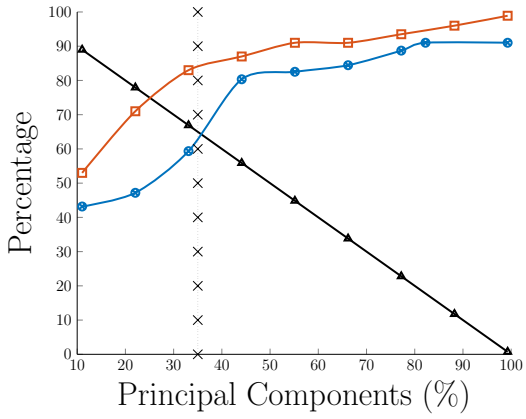
Table 9.5 shows how fast the Refine algorithm converges to the optimal values of $Umin$ and $Umax$ parameters relative to the brute force method. The brute force method checks all values from 0% to 100% and selects the optimal $Umin$ and $Umax$ that leads to the highest energy saving. Since the accuracy of the prediction is not 100%, the Refine heuristic improves the predicted values to reach the optimal value with few numbers of steps. The speedup for traffic ranging from 10% to 90% traffic volume is 14.85X to 25.71X of the brute force method. Similar to the accuracy of the predictor, the speedup of the Refine heuristics is independent of the traffic volume.



(a) Abilene

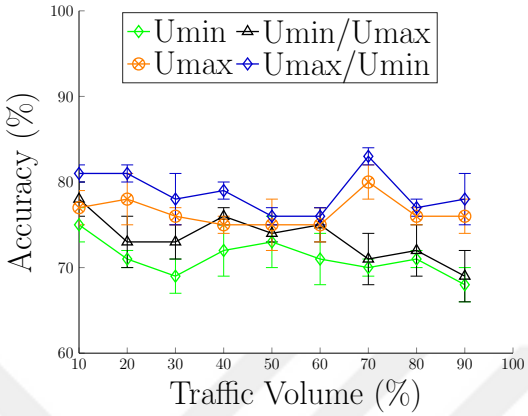


(b) GEANT

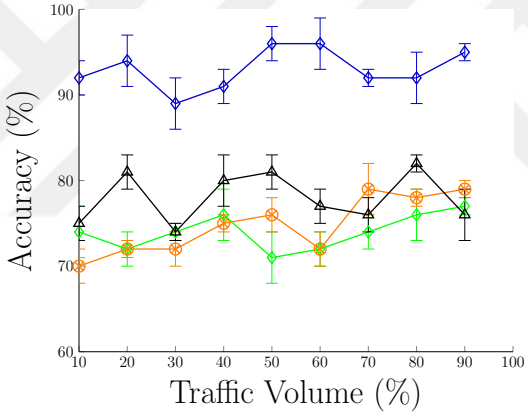


(c) Nobel-Germany

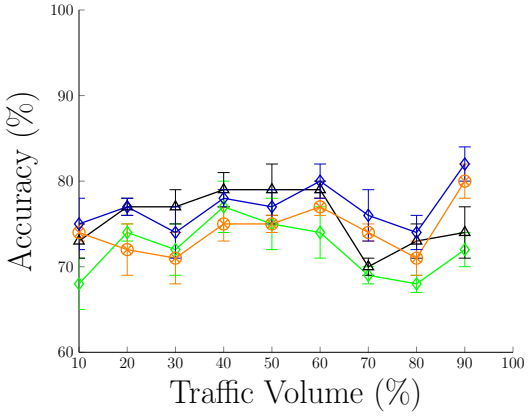
Figure 9.2: Cross validation results of the percentage of PCs versus feature size reduction, accuracy, and variance for Abilene, GEANT, and Nobel-Germany topology and traces. It also shows the optimal percentage of PCs selected for each topology and trace.



(a) Abilene



(b) GEANT

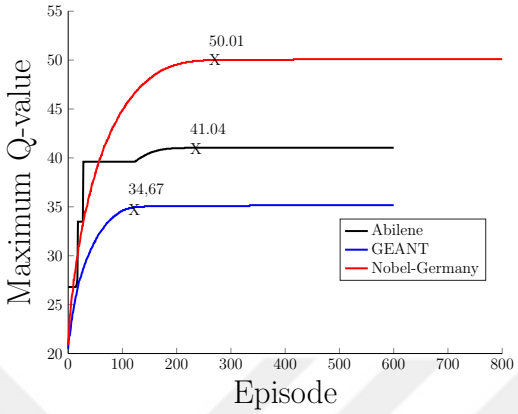


(c) Nobel-Germany

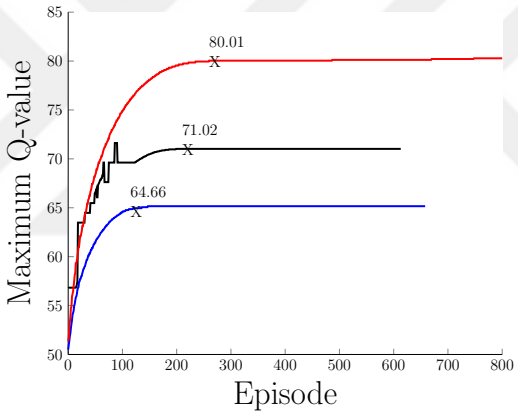
Figure 9.3: Accuracy for predicting $Umin$, $Umax$, $Umin/Umax$ ($Umin$ given $Umax$ is known, and $Umin/Umax$ ($Umax$ given $Umin$ is known a) Abilene b) GEANT and c) Nobel-Germany topology traces

9.4 Results for Reinforcement Learning Component of HyMER

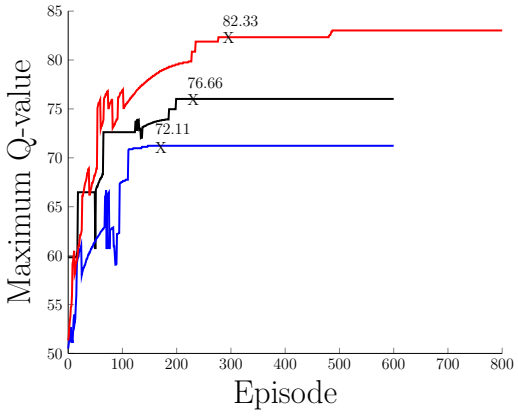
Figure 9.4 demonstrates the convergence of our proposed Q-Routing algorithm with regard to MaxR value. For MaxR value of 20, results show that the maximum Q-value of the algorithm is attained at the 215^{th}, 101^{th}, and 250^{th} iteration (episode) with maximum Q-value of 41.04, 34.67 and 50.01 for the Abilene, GEANT, and Nobel-Germany datasets, respectively. For MaxR=50, the maximum Q-value of 71.02, 64.66 and 80.01 is achieved at the 210^{th}, 120^{th}, and 275^{th} episodes similar topologies and traffic traces. For MaxR value of 70, the graph shows a fluctuation until it reaches the maximum Q-value. In all the three network topologies used and different MaxR values set in these experiments, the algorithm converges prior to reaching the maximum number of episodes (1000). Perhaps an interesting observation we see from these results is that the algorithm tends to exploit the maximum reward it has acquired after it reaches to some point. It is evident that exploration of better possibilities is done in prior episodes, then exploitation continues. The figure also shows that increasing the maximum reward MaxR value has a direct effect on the maximum Q-value, but the convergence of the algorithms remains similar in both cases.



(a) MaxR=20



(b) MaxR=50



(c) MaxR=70

Figure 9.4: Q-Routing maximum Q-value versus episodes for the Abilene, GEANT, and Nobel-Germany network topology and traffic traces by setting the learning rate α= 0.08, reward discount parameter γ= 0.1 and the number of episodes to 1000 a) MaxR=20, b) MaxR=50 c) MaxR=70

9.5 Energy Efficiency and Network Performance Results

This subsection presents energy efficiency and network performance experimental results. Percentage of links saved and the average power consumption of switches are the metrics used to measure network energy efficiency. The metrics presented to measure the network performance are average path length, throughput, and delay.

Figure 9.5 shows the percentage of links saved for Abilene, GEANT, and Nobel-Germany network topology and traffic traces. In the case of Abilene and Nobel-Germany topology, our approach HyMER achieves the maximum percentage of links saved. For traffic volume more than 70%, the B heuristics exhibits a closer saving to our approach for both network topologies. Especially, for the GEANT topology, the B heuristics demonstrated a slight improvement over our approach. There is a similar trend between NSP and NMU, because they are both first initialized with the shortest path and then flows are redirected in the direction of the next shortest path and next maximum utility, respectively. The figure demonstrates that our approach achieves the highest energy saving for lower traffic volumes. This clearly shows that the energy consumption of our approach is proportional to traffic volume streaming through the network. Hence, it achieves traffic proportionality as a feature in energy saving.

Figure 9.6 shows the average power consumption of the NEC PF5240 switches. Our approach HyMER demonstrates the least power consumption which saves a minimum of 10 watts to a maximum of 15 watts for the GEANT topology, 3 watts to 8 watts for the Abilene topology, and 2 watts to 5 watts for the Nobel-Germany topology. SPL and SDF heuristics show the least power saving. Our approach demonstrated similar results in Figure 9.5 where the power consumption increases with traffic volume. Hence, traffic proportional energy saving is achieved with HyMER similar to the MaxRESDN [15] heuristics.

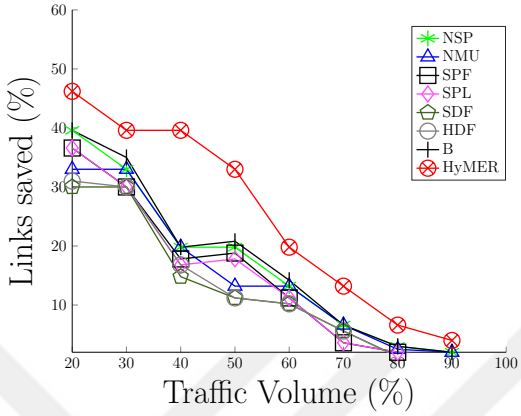
Figure 9.7 shows the average path length of our approach in comparison to other

heuristics for traffic volumes ranging from 20% to 90%. The NSP algorithm shows the lowest average path length. This is due to the fact that NSP gives priority to performance as it saves energy. SPL has demonstrated the worst average path length for Abilene and GEANT topology. HDF also has a similar trend with SPL. The reason for HDF and SPL to behave in this way is because their objective is energy saving but not performance. However, our approach, HyMER exhibits a very close average path length to NSP. By looking at Figures 9.5, 9.6, and 9.7, our approach has the highest energy saving and a performance closer to the best algorithm, NSP in terms of average path length. Hence, our approach maintains the trade-off between network performance and energy efficiency.

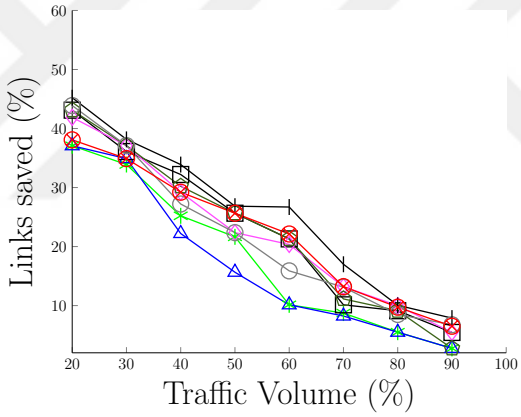
Figure 9.8 shows the throughput of our approach in comparison with other heuristics for traffic volumes ranging from 20% to 90%. NSP, NMU, and SPF have demonstrated the maximum throughput for all the topologies. This is mainly because the three heuristics give priority to performance as they try to save energy. Our approach HyMER has shown an acceptable throughput which is on average less than 3 to 5 Mbps from the best performing heuristics. The average path length measurements show that increase in traffic increases the average path length. This is because as the volume of traffic increases, the shortest paths become overloaded, and some flows have to be re-routed to longer paths. Our approach not only does have the maximum energy saving in terms of the average number of links saved and the average switch power consumption but also have a performance closer to performance-oriented energy saving heuristics. Hence, it maintains the trade-off between performance and energy efficiency.

Figure 9.9 shows the delay of our approach in comparison to other heuristics for traffic volumes ranging from 20% to 90% for the Abilene, GEANT and Nobel-Germany network topology and traffic traces. Similar to the results of throughput in Figure 9.8, heuristics such as NSP, SPF, and NMU that give priority to performance while saving energy achieve a delay of 9 ms for traffic volume of 20% and 11 ms for traffic volume of 90%. HDF and SDF heuristics on the other hand are better in energy

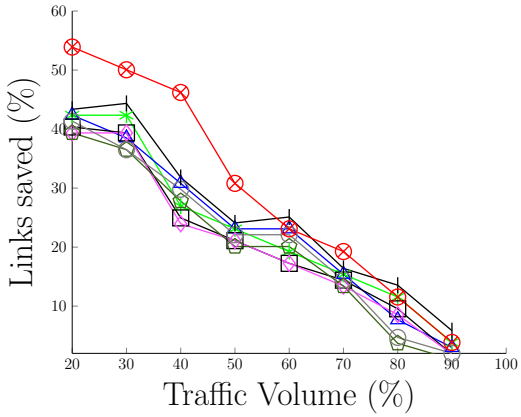
saving but exhibit 3 ms to 10 ms worse delay than NSP, NMU, and SPF algorithms. However, HyMER demonstrates a delay between the two types of heuristics. Results show that our approach is the best in terms of energy saving at the same time with acceptable delay. Hence, the trade-off between performance and energy efficiency is maintained.



(a) Abilene

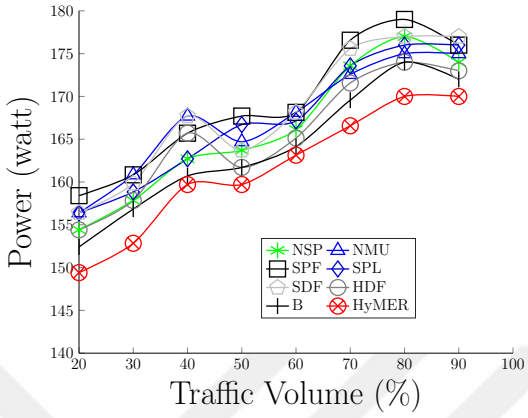


(b) GEANT

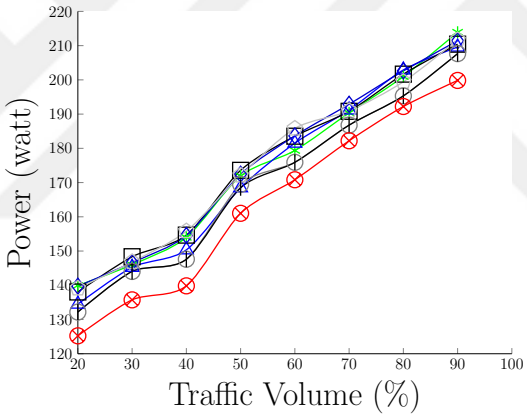


(c) Nobel-Germany

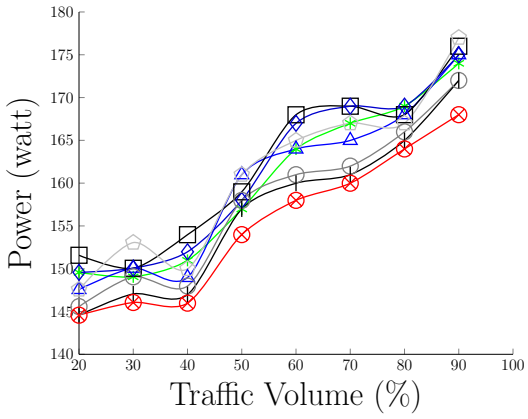
Figure 9.5: Percentage of links saved for traffic volume ranging from 20% to 90% for a) Abilene b) GEANT and c) Nobel-Germany network topology and traffic traces



(a) Abilene

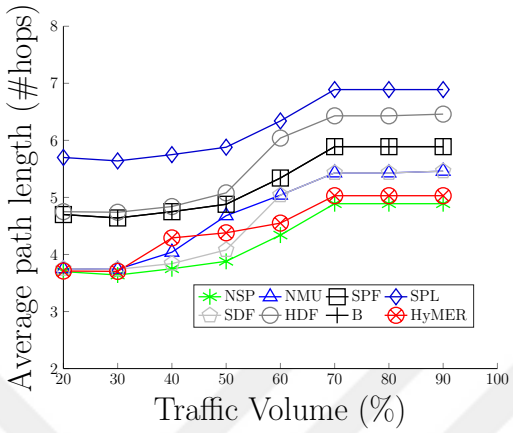


(b) GEANT

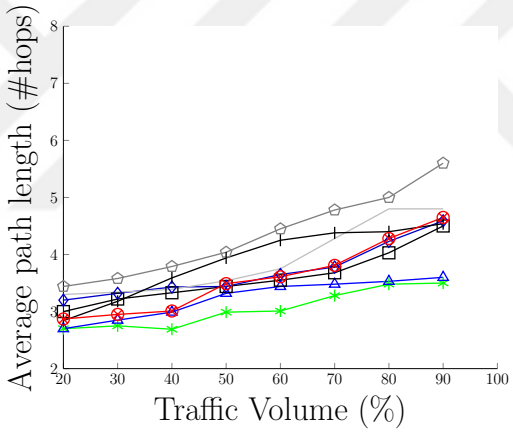


(c) Nobel-Germany

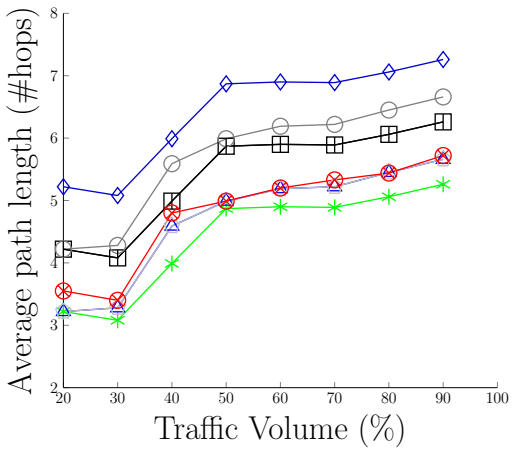
Figure 9.6: Average power consumption of the NEC PF5240 measured in watts for traffic volumes ranging from 20% to 90% for a) Abilene b) GEANT and c) Nobel-Germany network topology and traffic traces



(a) Abilene

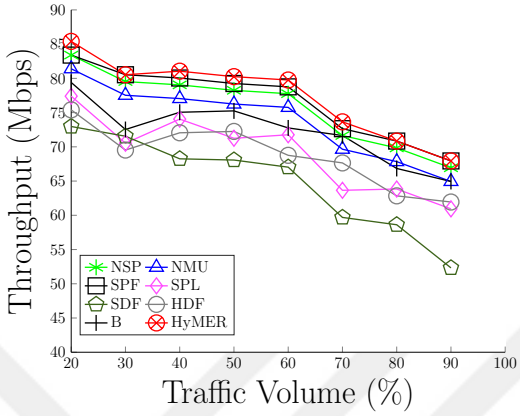


(b) GEANT

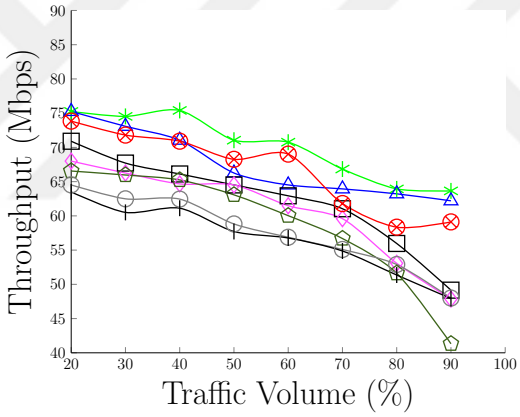


(c) Nobel-Germany

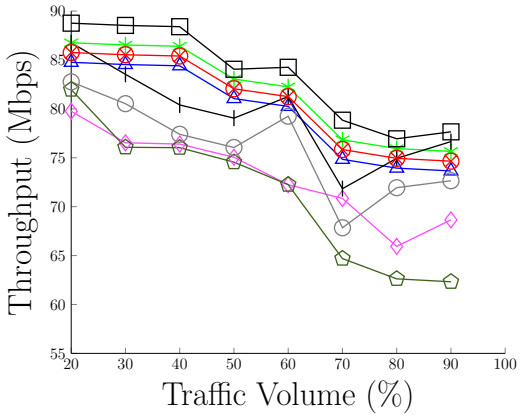
Figure 9.7: Average path length in terms of number of hops for traffic volumes ranging from 20% to 90% for a) Abilene b) GEANT and c) Nobel-Germany network topology and traffic traces



(a) Abilene

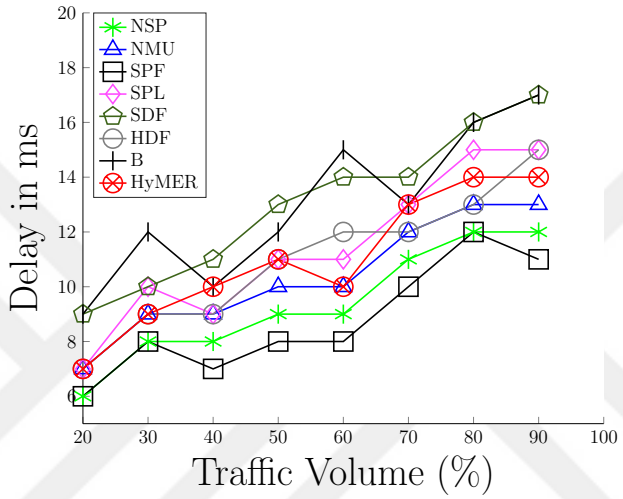


(b) GEANT

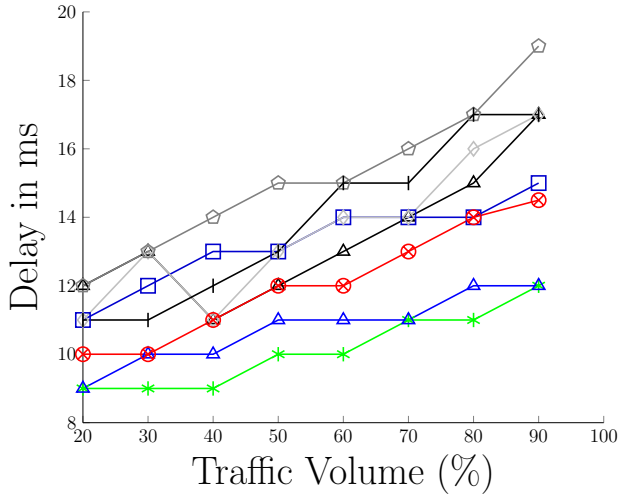


(c) Nobel-Germany

Figure 9.8: Throughput in Mbps for traffic volumes ranging from 20% to 90% for a) Abilene b) GEANT and c) Nobel-Germany network topology and traffic traces



(a) Abilene



(b) GEANT

Figure 9.9: Delay in ms for traffic volumes ranging from 20% to 90% for a) Abilene b) GEANT network topology and traffic traces

9.6 Discussion and Applicable Scenarios

The major objectives in the design of HyMER are fourfold; energy efficiency, network performance, dynamicity, and computational feasibility. Figure 9.5 and 9.6 illustrate the energy efficiency and average path length of the supervised component of HyMER. According to the predicted $Umin$ and $Umax$ parameters and after applying the Refine heuristics, results show that HyMER heuristics achieves energy saving of 48% for low traffic in terms of percentage of links saved. Switch power consumption has shown on average 10 watts less than heuristics that give priority to performance.

Figures 9.7, 9.8, and 9.9 show the average path length, throughput, and delay of HyMER in comparison with similar approaches. It is demonstrated that energy saving approaches with the objective of increasing performance in terms of number hops, such as NSP, NMU, and SPF, have shown the least average path length, the highest throughput, and the least delay. Approaches that give more attention to energy saving, such as HDF and SPL, have demonstrated the opposite. Our approach, however, shows a performance closer to approaches that give higher priority to performance over energy efficiency.

The significance of the supervised component of the HyMER lies in the fact that it predicts $Umin$ and $Umax$ for new traffic. Getting the optimal values of the parameters increases the RESDN value. Maximum RESDN value increases utilities of links, energy saving, and also maintains an acceptable network performance. However, there are two disadvantages of the supervised approach. First, there needs to be a sufficient amount of data for model training, otherwise the model overfits and would not generalize well for new data. Second, the supervised component is applied by taking the snapshots of the network state periodically, but not in response to the dynamic changes in the network environment. The supervised component fails to capture the network state if major changes happen between two consecutive time intervals that snapshots are taken. Setting small time intervals between snapshots, on the other hand, incurs extra prediction costs especially if there is no major change in the network.

The reinforcement component addresses the drawbacks of the supervised approach. First, the reinforcement model can be trained by interacting with the network environment and traffic which makes it applicable despite insufficient amount of training data. Second, it can automatically learn the changes in the network and traffic. Hence, it finds energy efficient paths for newly arriving flows and at the same time considers the changes in the topology. On the other hand, applying reinforcement learning from scratch, comes with extra cost of time. However, the reinforcement component can be applied on the top of the results from the supervised component, in which case the convergence time of the reinforcement algorithm drastically decreases.

Table 9.6: HyMER supervised and reinforcement components: Application scenarios

Component	**Sufficient Historical Data**	**From Scratch**	**Dynamicity Requirement**
Supervised (S)	✓		
Reinforcement (R)		✓	✓
Combined (S+R)	✓	✓	✓

Table 9.6 shows the application scenarios for the supervised and reinforcement components of the HyMER framework. It is our finding that applying both the supervised and reinforcement components not only would help to learn from historical data but also adapt quickly to the changing traffic and topology. It is demonstrated that when applied on the top of the results of the supervised solutions, the reinforcement component converges two times faster than running from scratch.

Chapter 10

End System Aware Energy Efficiency Using SDN

In data centers, multiple virtual machines are placed in physical servers, multiple physical servers in tern are placed in a rack. The physical machines are connected to each other using high speed switches, routers and links. It is common practice to aggregate hundreds to thousands of physical machines into a cluster for performance. Bcube, Fat tree, Dcell, and VL2 are popular architectures of organizing network components, physical servers, and racks. In Software Defined Data Centers (SDDCs), SDN controller is responsible managing the network components and the traffic flow. OpenStack, Apache CloudStack, and Google Compute Engine are among the few software tools that are used to manage computing, networking, and storage resources [121, 122].

The utility of physical servers is less than 35% in data centers [123]. The power consumption of an idle and 20% utilized server is 70% and 80% of their peak utility [124, 125]. Virtual machine placement and migration play a major role in minimizing the power consumption of servers. Energy saving virtual machine placement or migration methods focus on reducing the number of physical servers by moving virtual severs from underutilized physical machines. Virtual migration, sends the memories and storage contents of virtual machines from source to destination physical server.

This comes up with network overhead cost. Energy saving approaches fail to minimize the energy consumption due to networking components.

The work in [33, 36, 39] jointly minimizes the power consumption of physical machines and the traffic between virtual machines in SDN. To the best of our knowledge, we are the first to address the problem of minimizing the power consumption of physical machines and minimizing the power consumption of network components. Motivated by Ratio for Energy Saving in SDN (RESDN) [15], We propose a new energy efficiency metrics for software defined data centers called Ratio for Energy Saving of Physical Machines (RESPM) based on the utility of physical machines by virtual machines.

The contributions of this module are as follows.

- We propose an energy efficiency metric RESPM which is based on physical machines utility interval. We also extended our network component energy efficiency and performance metric RESDN [18] to be used jointly with RESPM. Unlike other approaches which focus on server energy efficiency or network component efficiency, our approach considers both simultaneously.

- We present an IP formulation with objective to jointly maximize the RESPM and RESDN.

- Experiments are conducted on real world data center virtual machine placement traces. Results show that our approach achieves up to to 9% better RESPM value, up to 35% better RESDN value, and more than 50% links saving as compared to state-of-the-art energy saving virtual machine migration algorithm.

10.1 Preliminaries of End System Aware Energy Efficiency in SDN

The goal of the end system aware solution is to minimize the number of active physical machines through migrating the virtual machines into fewer number of physical machines. Reviewing various models used to solve the problem, we propose a general model for the end system aware energy efficiency. End system aware solutions should be addressed as server consolidation and network optimization problems simultaneously [35, 36, 67]. The general model we propose addresses both problems and is inspired by [36].

Server consolidation and network optimization are the two components of virtual machine migration. The server consolidation problem is modeled as a quintuple ($\mathbb{P}$, $\mathbb{V}$, $\mathbb{R}p$, $\mathbb{R}v$, $\mathbb{R}$) where $\mathbb{P}$, $\mathbb{V}$, $\mathbb{R}p$, $\mathbb{R}v$, and $\mathbb{R}$ correspond to the set of physical machines, set of virtual machines, matrix of resources of physical machines, matrix of resource requirements of virtual machines, and vector of type of resources, respectively. The resources are listed as but not limited to CPU, memory and bandwidth capacities. The dimensions of the matrix $\mathbb{R}p$ and $\mathbb{R}v$ are $|\mathbb{P}|$ by $|\mathbb{R}|$ and $|\mathbb{V}|$ by $|\mathbb{R}|$, respectively.

The network optimization component of the virtual migration problem models the network as a graph as an undirected weighted graph $\mathbb{G}$= ($\mathbb{Z}$,$\mathbb{E}$) where $\mathbb{Z}$ is the set of switches and $Z_i \in \mathbb{Z}$ represents switch i and $e_{ij} \in \mathbb{E}$ represents that there is link between switches Z_i and Z_j. The weight W_{ij} corresponds to the bandwidth of the link connecting switches Z_i and Z_j.

Let X be a placement matrix where

$$X_{ij} = \begin{cases} 1, & \text{if virtual machine} j \text{ is placed on physical machine } i \\ 0, & \text{otherwise} \end{cases}$$

P_i^r is the amount of resource type r available on physical machine i, whereas V_j^r is the demand of virtual machine j for resource type r. Let PM_i is a binary variable

with value 1 if physical machine i is on, or 0 otherwise. The objective function of the general end system aware solution is the combination of the server consolidation and network optimization.

10.2 Motivation

Infrastructure as Service (IaS) provides the users with infrastructure services and the providers needs to meet certain Service Level Agreements (SLAs). SLA violations is a major cause for users to migrate from on provider to the other which makes the provider loose money. The providers hence keep servers and network components to operate with maximum capacity even if the demand by users is less. This incurs additional cost of electricity, computational, and networking resources. The problem that need to be tackled is to minimize the costs with out having to violate SLAs.

Let PMU be the energy utility of a physical machine P defined as

$$PMU_p = \begin{cases} 1, & \text{if } PU_{min} \leq U_p \leq PU_{max} \\ \sigma & \text{if } PU_{min} > U_p \ \& \ PU_{min} - U_p > \sigma \\ \sigma & \text{if } PU_{max} < U_p \ \& \ U_p - PU_{max} > \sigma \\ 0, & \text{otherwise} \end{cases} \tag{10.1}$$

where PU_{min}, PU_{max}, and σ are the minimum utility of a physical server, maximum utility of a physical server, and tolerance parameter respectively.

Ratio of Energy Saving for Physical Machines ($RESPM$) is calculated as follows.

$$RESPM = \frac{\sum_{\forall p \in \mathbb{P}} EU_p C_p}{\sum_{\forall p \in \mathbb{P}} Y_p C_p} \tag{10.2}$$

where C_p, EU_p, and Y_p are the total capacity, the energy utility value, and binary status of physical machine p respectively.

$$Y_p = \begin{cases} 1, & \text{if P} \in \mathbb{P} \text{ is active} \\ 0, & \text{otherwise} \end{cases} \tag{10.3}$$

Similarly, we introduce a metric for the efficiency of the network components in terms of energy saving and performance [15].

Let binary variable S_i denote the status of switch Z_i such that

$$S_i = \begin{cases} 1, & \text{if switch } Z_i \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

Let the traffic generated due to migrating memory in the network is represented by the set of flows $\mathbb{F}$ where $f \in \mathbb{F}$ is defined as $f{=}(sr, ds, \lambda_f)$. sr and $ds \in \mathbb{Z}$ are the source and destination switches of flow f, and λ_f is the rate of f measured in bits per second.

$$F_{ij} = \begin{cases} 1, & \text{if flow } \ f \text{ passes through edge } e_{ij} \\ 0, & \text{otherwise} \end{cases}$$

$\mathbb{U}$ is the set of the utilities of every edge in the graph $\mathbb{G}$ where $U_{ij} \in \mathbb{U}$

$$U_{ij} = \frac{\sum_{\forall f} F_{ij} x \lambda_f}{W_{ij}} \tag{10.4}$$

is defined as the ratio of the sum of the rates of the flows passing through the edge e_{ij} to the link bandwidth W_{ij}. Utility of a link is between 0 and 1, where 0 means no flow is passing through the link and 1 means the sum of the flow rates passing through the link is equal to the link bandwidth. Let $Umin$ be the minimum utility value to keep a link active and $Umax$ is the maximum utility of a link.

$$X_{ij} = \begin{cases} 1, & Umin \leq U_{ij} \leq Umax \\ 0, & \text{otherwise} \end{cases}$$

The network component energy efficiency metric is formally defined as follows.

$$RESDN = \frac{\sum_{\forall e_{ij}} X_{ij} W_{ij}}{\sum_{\forall e_{ij}} L_{ij} W_{ij}} \tag{10.5}$$

where W_{ij} is the band width of edge e_{ij}, L_{ij} is binary variable that denotes the status of edge e_{ij}

$$L_{ij} = \begin{cases} 1, & \text{if edge } e_{ij} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

10.3 IP Formulation for jointly maximizing RESPM and RESDN

$$\text{maximize } \gamma RESPM + \beta RESDN \tag{10.6}$$

$$\text{subject to } \sum_{j=1} V_j^r * X_{ij} \leq P_i^r \quad \forall i, r \tag{10.7}$$

$$\sum_i X_{ij} = 1 \ \ \forall j \tag{10.8}$$

$$PM_i \geq X_{ij} \ \ \forall i, j \tag{10.9}$$

$$\sum_{\forall f} F_{ij} x \lambda_f \leq W_{ij} \ , \forall e_{ij} \tag{10.10}$$

$$\sum_{\forall f} F_{ai} = \sum_{\forall f} F_{ib} \ , \ Z_i \neq f^{sr}, Z_i \neq f^{ds} \tag{10.11}$$

$$F_{mj} = F_{in} \ , \forall f \ Z_m = f^{sr} \ , \ Z_n = f^{ds} \ , \forall e_{mj} \ , \exists e_{in} \tag{10.12}$$

$$F_{ij} \leq S_j \ and \ F_{ji} \leq S_j \ , \forall Z_j \in \mathbb{Z} \tag{10.13}$$

$$S_i \leq \sum_{\forall f} [F_{ij} + F_{ji}] \ , \ \forall Z_i \in \mathbb{Z} \tag{10.14}$$

$$L_{ij} \leq S_i \text{ and } L_{ij} \leq S_j \ \forall Z_i, Z_j \in \mathbb{Z} \tag{10.15}$$

$$\tag{10.16}$$

The objective function (equation 10.6) jointly maximizes the RESPM and RESDN values of a software defined data center. The constraint 10.7 states the sum of resource demands of virtual machines installed on a given physical machine cannot be more than the capacity of the physical machine. Constraint 10.8 limits that each virtual machine can be placed on exactly one physical machine. Equation 10.9 associates the variables PM_i and X_{ij} by asserting that a physical machine will be turned on or off depending on whether it is used or not.

The constraint in Equation 10.10 states that the sum of the rates of flows between two switches should not exceed the link capacity. The constraint in Equation 10.11 states that the number of flows entering and leaving switches which are neither destin-

ation nor sources of flow should be equal. The constraint in Equation 10.12 assures a flow entering from source switch should reach the destination switch. The constraint in Equation 10.13 asserts that a flow should not be assigned to a link that is connected to an inactive switch. Constraint in Equation 10.14 models the relationship between the flows passing through a link and a switch. It asserts that if no flow is passing through all the links connected to switch Z_i, then change the binary variable $S_i = 0$. The constraint in Equation 10.15 asserts that a link should be put in active state if and only if both of the switches it is connecting are active, otherwise, it is inactive.

10.4 Framework for End System Aware Energy Efficiency in SDN

The objective of the end system aware framework is to jointly minimize the energy consumption of end systems and network components. Figure 10.1 demonstrates the framework with its components designed for end system aware energy efficiency using SDN. Traffic manager, topology manager, EA optimizer, and end system manager are the four components of the framework.

The traffic generated due to virtual machine migration is passed to the traffic manager. The traffic manager transfers the traffic information to the EA optimizer. The optimizer generates an optimal subgraph based on the traffic volume, the utility of the links according to the migrating virtual machines. Low traffic load should result in a subgraph with smaller number of active links and switches, whereas high traffic load should increase the number of active links and switches in the subgraph respectively. In the former, the sleep/turn-off decision for a subset links and switches would be achieved. In the latter, however, a subset of links and switches would be turned on or made active. The end system manager is responsible for deciding the virtual machines that have to be migrated and also managing the actual migration.

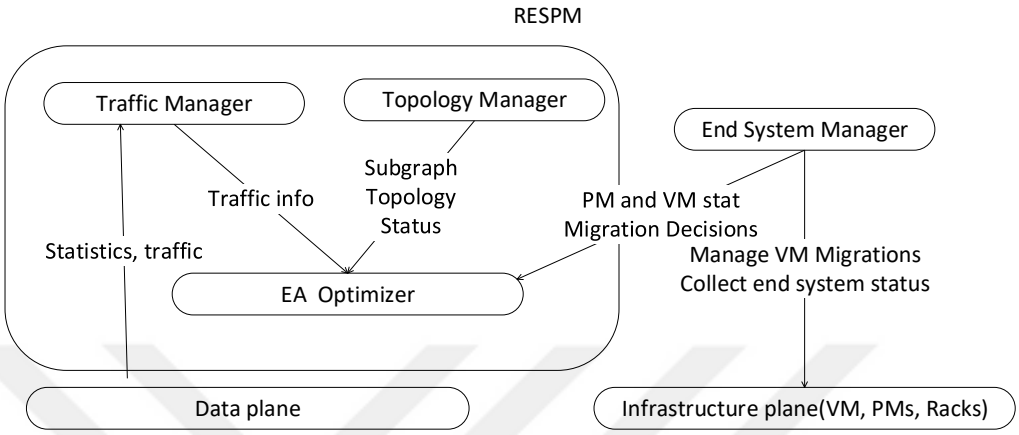


Figure 10.1: Framework for end system aware energy efficiency using SDN

10.5 Experimental Results

The experiments are conducted on real-world data center data pulled for 25 days. The total number of physical machines is 128 and the total number of virtual machines is 3482. The average number of virtual machines that are running per day is on average 1112. The physical machines are powered on all the time. Figure 10.2 shows the histogram of the average utility of physical machines for 25 days. It shows that the average utility of the physical machines is less than 35% and more than 40% of physical machines are utilized less than 10%.

The topology structure that is used to connect the physical servers is fat-tree. Table 10.1 shows the characteristics of the network topology.

Table 10.1: The characteristics of the data center topology

Fat-tree Property	**Values**
K	10
#Core switches	25
#Aggregate switches	55
#Edge switches	55
Maximum # of servers	250
#Links (Core - Aggregate switch)	125
#Links(Aggregate - Edge switches)	275
#Links (Edge switches - Servers)	275

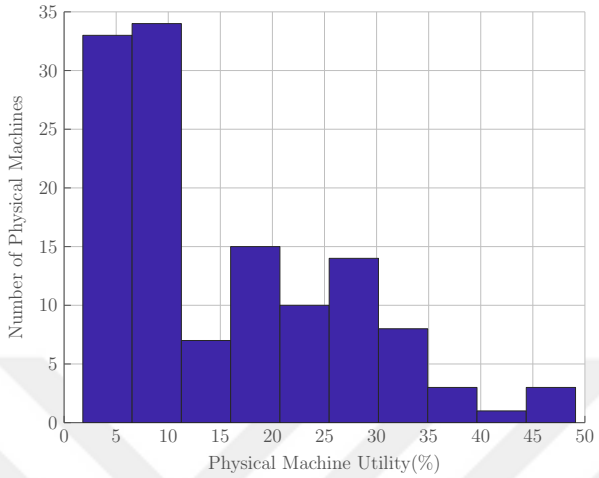


Figure 10.2: Histogram of average utilities of physical machines for 25 days

Table 10.2 shows the metrics we used in the experiments, that are are RESPM, RESDN, and number of physical machines turned off.

Table 10.2: Metrics Used in this experiment

Metrics	**Meaning**	**Unit**
RESPM	Ratio for Energy Saving of Physical Machines	%
RESDN	Ratio for Energy Saving in SDN	%
Physical Machines	Percentage of physical machines powered off	%

We compare our approach with state-of-the art Power-aware and Performance-guaranteed Virtual Machine Placement (PPVMP) algorithm [126]. The PPVMP algorithm attempts to save the power consumption of physical machines and guarantee acceptable migration performance. Figure 10.3 shows the percentage of physical machines powered off after the virtual machines are migrated. It is depicted that our approach saves 2 to 4 physical machines better than PPVMP.

Figure 10.4 shows the RESPM value of the data center for MaxRESPM. Our approach has up to 9% better in RESPM value as compared to a PPVMP.

Our approach exhibits more than 50% links saving and 35% better RESDN value as compared to PPVMP. These initial results show that our approach has the potential

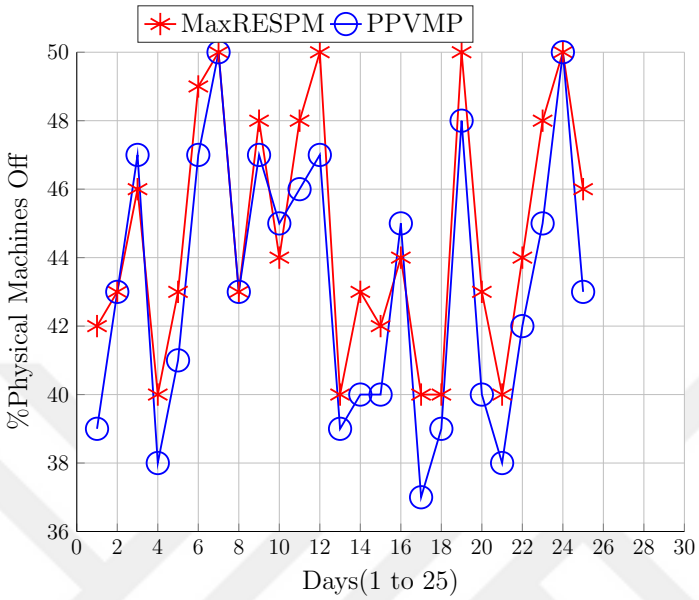


Figure 10.3: The average utilities of physical machines for 25 days

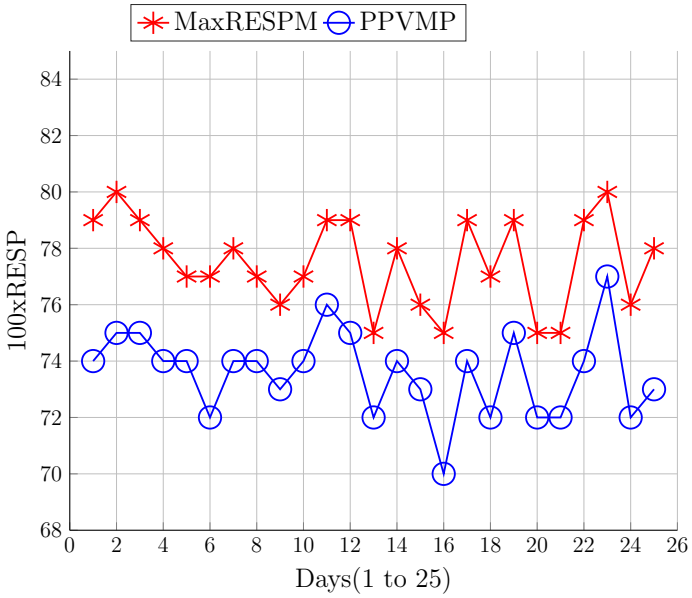


Figure 10.4: RESPM values for a VM trace of 25 days

to minimize the power consumption of physical machines and network components simultaneously. As future direction, the end system aware module of the controller is aimed to be further developed to incorporate migration time, down time, CPU utility, and other relevant metrics.

Chapter 11

Conclusion

11.1 Remarks

The flexibility of network control by SDN can be used for energy efficient routing. However, minimizing energy consuming devices without compromising performance is a challenging task. We proposed energy efficiency metric named RESDN (Ratio for Energy Saving in SDN) that puts the utility of links under consideration. Unlike other energy efficiency metrics, RESDN is based on the link utility interval parameters, and captures the dynamic network changes in SDN. We also provided an IP formulation with the objective of increasing the RESDN of a network environment and a heuristics named MaxRESDN that maximizes energy saving. The experiments are conducted on Mininet network emulator and POX controller using the GEANT network topology and dynamic traffic traces. We also simulated the power consumption of OvSwitch, NEC PF5240, and Zodiac FX OpenFlow switches.

Experimental results show that maximizing the RESDN value improves energy efficiency while maintaining acceptable network performance. MaxRESDN is up to 30% better in the number of links and achieves up to 14.7 watts, 10 watts, and 3.2 watts less power consumption for NEC, OVS and Zodiac FX switches respectively as compared to other utility-based heuristics for energy efficient routing. The maximum RESDN value is achieved by our proposed MaxRESDN method, which also performs

close to the solutions that give priority to network performance in terms of average path length, throughput, and delay. Our approach not only does have 3 to 5% higher traffic proportionality but also maintains the trade-off between network performance and energy efficiency. Since the performance of the MaxRESDN heuristics greatly depends on the value of utility interval parameters, we performed a detailed analysis of the parameters with regard to traffic volume. It is demonstrated that the values of utility interval parameters directly affect the efficiency of the algorithm.

In this thesis, we proposed HyMER, supervised and reinforcement machine learning based framework for traffic aware energy efficient routing in SDN. We used topology and traffic as features to train our model. For the supervised component of the framework, we employed PCA and achieved a feature size reduction of more than 65% on real-world network topology and dynamic traffic traces. Particularly, we tested supervised component to predict $Umin$ and $Umax$ parameters for the energy efficient MaxRESDN heuristics algorithm. In addition to the prediction, we also proposed a heuristics to increase the accuracy of the supervised model prediction to 100%. Experiment results show that the accuracy of predicting $Umin$ and $Umax$ is more than 70%. The refining heuristics algorithm converges to the optimal $Umin$ and $Umax$ values 15 to 25 times faster than the brute force method.

The supervised component is applicable when there is sufficient historical data to learn from. The reinforcement component addresses the two problems of the supervised component. First, there must be sufficient amount of historical data. Second, they cannot capture the dynamic nature of the network. The reinforcement learning component of the framework incrementally learns by interacting with the network environment in picking the path that maximizes energy saving. It can work in both cases where historical data is available or not. Experiments conducted on real network topology and traffic traces show that, it converges to the MaxRESDN energy saving with an average 110 to 270 number of episodes for different values of MaxR. It also converges even faster when it is initialized by outputs of the supervised component.

As initial results of the end system aware module of the controller, we proposed an

physical server utility interval based metric with an objective of minimizing number of physical machines and keep the performance of the physical machine at acceptable range. According to initial results, compared with state-of-the-art energy efficient VM migration algorithm, our approach demonstrated up to 9% better physical machine saving and more than 50% links saving.

11.2 Future Directions

As future work, we aim to to apply the idea behind RESDN metric to incorporate switches and improve the scalability through parallelizing the MaxRESDN algorithm to handle multiple flow arrivals concurrently. The HyMER framework can be extended to incorporate end systems by using SDN as a tool for energy efficient and performance oriented virtual machine placement and migration planning in data centers. Application of the machine learning framework in edge computing [127], wireless networks [128], and software defined vehicular networks [129] can be promising future directions.

In this thesis, we have shown how reinforcement learning is applied on top of the supervised component outcome. An interesting future work could be enabling the controller to automatically decide when to switch from the supervised component usage to the reinforcement component. Another research direction is analyzing the fault tolerance of HyMER in the case of link and switch failures.

The rule placement mechanisms affect the network performance and also determines the routing. Forwarding rules are determined and pushed to the switches by the controller. Thus, rule space optimization in SDN can be formulated as a global network-level problem. Placing the rules to respective switches distributed across the network and optimizing an objective function under the constraints is NP-hard problem. Given a routing policy and end-point policy of the network, solutions for space efficient rule representation that would lead to energy savings are necessary. Energy efficient solutions in this group need to formalize the energy cost model and the con-

straints associated, then apply heuristic algorithms to find optimum energy saving strategy. As another future direction, the general optimization model we presented for rule placement solutions would provide guidelines for the objective functions, parameters and constraints to be considered [5].

Bibliography

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] K. Diego *et al.*, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan 2015.

[3] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 27–51, Firstquarter 2015.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] B. G. Assefa and Ö. Özkasap, "A survey of energy efficiency in sdn: Software-based methods and optimization models," *Journal of Network and Computer Applications*, vol. 137, pp. 127–143, 2019.

[6] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 463–476, April 2012.

[7] S. Lambert, W. Van Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet, "Worldwide energy needs for ICT: The rise of power-aware networking," in *Advanced Networks and Telecommunication Systems (ANTS), 2008. 2nd International Symposium on*, pp. 1–3, Dec 2008.

[8] B. G. Assefa and O. Ozkasap, "State-of-the-art energy efficiency approaches in software defined networking," in *ICN (The Fourteenth International Conference on Networks), Symposium on Advances in Software Defined Networks (SoftNetworking)*, pp. 256–260, 2015.

[9] GreenPeace, "Clicking clean: How the Companies are creating the green internet," 2014.

[10] S. Brienza, S. E. Cebeci, S. S. Masoumzadeh, H. Hlavacs, O. Özkasap, and G. Anastasi, "A survey on energy efficiency in p2p systems: File distribution, content streaming, and epidemics," *ACM Computing Surveys*, vol. 48, no. 3, 2016.

[11] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, 2016.

[12] A. Hammadi and L. Mhamdi, "A Survey on Architectures and Energy Efficiency in Data Center Networks," *Computer Communications*, vol. 40, pp. 1 – 21, 2014.

[13] R. Soua and P. Minet, "A survey on energy efficient techniques in wireless sensor networks," in *Wireless and Mobile Networking Conference (WMNC), 2011, 4th Joint IFIP*, pp. 1–9, Oct 2011.

[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.

[15] B. G. Assefa and Ö. Özkasap, "RESDN: A novel metric and method for energy efficient routing in software defined networks," *IEEE Transactions on Network and Service Management, Under Review*, vol. abs/1905.12219, 2019.

[16] B. G. Assefa and O. Ozkasap, "Framework for traffic proportional energy efficiency in software defined networks," *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2018.

[17] B. G. Assefa and O. Ozkasap, "Link utility and traffic aware energy saving in software defined networks," *2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2017.

[18] B. G. Assefa and O. Ozkasap, "A novel utility based metric and routing for energy efficiency in software defined networking," in *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–4, June 2019.

[19] B. G. Assefa and Ö. Özkasap, "Mer-sdn: Machine learning framework for traffic aware energy efficient routing in sdn," *IEEE Intl Conf on Big Data Intelligence and Computing (DataCom)*, pp. 974–980, 2018.

[20] B. G. Assefa and Ö. Özkasap, "Hymer: A hybrid machine learning framework for energy efficient routing in sdn," *IEEE Journal on Selected Areas in Communications, Series on Network Softwarization & Enablers, Under Review*, 2019.

[21] N. PF540, "NEC ProgrammableFlow PF5240 Switch." `https://www.necam.com/PFlow/doc.cfm?t=PFlowPF5240Switch`, 2018. Accessed: 2018-12-26.

[22] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks," in *NSDI*, 2010.

[23] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," in *Local Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pp. 1–6, Oct 2011.

[24] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford, "Identifying and Using Energy-critical Paths," in *Seventh Conference on Emerging Networking EXperiments and Technologies*, CoNEXT, 2011, pp. 18:1–18:12, ACM, 2011.

[25] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "CARPO: Correlation-AwaRe power optimization in data center networks," in *IEEE INFOCOM, March 25-30, 2012*, pp. 1125–1133, 2012.

[26] N. H. Thanh, P. N. Nam, T.-H. Truong, N. T. Hung, L. K. Doanh, and R. Pries, "Enabling experiments for energy-efficient data center networks on openflow-based platform," in *Communications and Electronics (ICCE), 2012 Fourth International Conference on*, pp. 239–244, IEEE, 2012.

[27] T. Nguyen *et al.*, "Modeling and Experimenting Combined Smart Sleep and Power Scaling Algorithms in Energy-aware Data Center Networks ," *Simulation Modelling Practice and Theory*, vol. 39, pp. 20 – 40, 2013.

[28] R. Wang, Z. Jiang, S. Gao, W. Yang, Y. Xia, and M. Zhu, "Energy-aware routing algorithms in software-defined networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pp. 1–6, June 2014.

[29] T. H. Vu, V. C. Luc, N. T. Quan, T. M. Nam, N. H. Thanh, and P. N. Nam, "The new method to save energy for Openflow Switch based on traffic engineering," in *Electronic Design (ICED), 2014 2nd International Conference on*, pp. 309–314, Aug 2014.

[30] T. H. Vu, V. C. Luc, N. T. Quan, N. H. Thanh, and P. N. Nam, "Energy saving for OpenFlow switch on the NetFPGA platform based on queue engineering," *SpringerPlus*, vol. 4, no. 1, p. 64, 2015.

[31] F. Giroire, J. Moulierac, T. K. Phan, and F. Roudaut, "Minimization of network power consumption with redundancy elimination," in *Computer communications*, vol. 59, pp. 98–105, Elsevier, 2015.

[32] B. B. Rodrigues, A. C. Riekstin, G. C. Januário, V. T. Nascimento, T. C. Carvalho, and C. Meirosu, "Greensdn: Bringing energy efficiency to an sdn emulation environment," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 948–953, May 2015.

[33] H. Zhu, X. Liao, C. de Laat, and P. Grosso, "Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform," *Journal of Network and Computer Applications*, vol. 63, pp. 110 – 124, 2016.

[34] R. Ana C, J. Guilherme C., R. Bruno B, N. Viviane T, C. Tereza C.M.B., and M. Catalin, "Orchestration of energy efficiency capabilities in networks," *Journal of Network and Computer Applications*, vol. 59, pp. 74 – 87, 2016.

[35] H. Shirayanagi, H. Yamada, and K. Kenji, "Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks," in *Computers and Communications (ISCC), IEEE Symposium on*, pp. 460–467, July 2012.

[36] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*, pp. 2876–2880, IEEE, 2012.

[37] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, and N. Pissinou, "Joint host-network optimization for energy-efficient data center networking," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pp. 623–634, IEEE, 2013.

[38] S.-H. Wang, P. P.-W. Huang, C. H.-P. Wen, and L.-C. Wang, "EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks," in *Information Networking (ICOIN), International Conference on*, pp. 220–225, Feb 2014.

[39] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *INFOCOM, 2014 Proceedings IEEE*, pp. 2598–2606, IEEE, 2014.

[40] L. Chen, J. Zhang, L. Cai, R. Li, T. He, and T. Meng, "Mtad: a multitarget heuristic algorithm for virtual machine placement," *International Journal of Distributed Sensor Networks*, vol. 2015, p. 5, 2015.

[41] A. Dalvandi, M. Gurusamy, and K. C. Chua, "Power-efficient resource-guaranteed vm placement and routing for time-aware data center applications," *Computer Networks*, vol. 88, pp. 249–268, 2015.

[42] J. Ortigoza, F. Lopez-Pires, and B. Barán, "Dynamic environments for virtual machine placement considering elasticity and overbooking," *arXiv preprint arXiv:1601.01881*, 2016.

[43] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the "One Big Switch" Abstraction in Software-defined Networks," in *Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT, (New York, NY, USA), pp. 13–24, ACM, 2013.

[44] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *IEEE INFOCOM*, pp. 545–549, April 2013.

[45] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 2523–2529, IEEE, Dec 2014.

[46] F. Giroire, F. Havet, and J. Moulierac, "Compressing two-dimensional routing tables with order," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 351–358, 2016.

[47] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "OFFICER: A general Optimization Framework for OpenFlow Rule Allocation and Endpoint Policy Enforcement," in *The 34th Annual IEEE International Conference on Computer Communications (INFOCOM 2015)*, (Hongkong, China), Apr. 2015.

[48] M. Rifai, N. Huin, C. Caillouet, F. Giroire, D. M. L. Pacheco, J. Moulierac, and G. Urvoy-Keller, "Too many SDN rules? Compress them with MINNIE," in *GLOBECOM*, 2015.

[49] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[50] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[51] S. A. Pistirica, O. Poncea, and M. C. Caraman, "QCN Based Dynamically Load Balancing: QCN Weighted Flow Queue Ranking," in *Control Systems and Computer Science (CSCS), 2015 20th International Conference on*, pp. 197–204, IEEE, 2015.

[52] W. Rui, SuixiangGao, Y. Wenguo, and J. Zhipeng, "Bandwidth-aware energy efficient routing with sdn in data center networks," in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pp. 766–771, IEEE, 2015.

[53] A. Markiewicz, P. N. Tran, and A. Timm-Giel, "Energy consumption optimization for software defined networks considering dynamic traffic," in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pp. 155–160, Oct 2014.

[54] Z. Guo, S. Hui, Y. Xu, and H. J. Chao, "Dynamic flow scheduling for power-efficient data center networks," in *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, pp. 1–10, IEEE, 2016.

[55] M. Rahnamay-Naeini, S. S. Baidya, E. Siavashi, and N. Ghani, "A traffic and resource-aware energy-saving mechanism in software defined networks," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, IEEE, 2016.

[56] A. Fernandez-Fernandez, C. Cervello-Pastor, and L. Ochoa-Aday, "Achieving energy efficiency: An energy-aware approach in sdn," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Dec 2016.

[57] A. Fernández Fernández *et al.*, *Energy-aware routing techniques for software-defined networks*. PhD thesis, Universitat Politècnica de Catalunya, 2018.

[58] A. Fernández-Fernández, C. Cervelló-Pastor, and L. Ochoa-Aday, "Energy efficiency and network performance: A reality check in sdn-based 5g systems," *Energies*, vol. 10, no. 12, p. 2132, 2017.

[59] J. Luo, S. Zhang, L. Yin, and Y. Guo, "Dynamic flow scheduling for power optimization of data center networks," in *Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 57–62, Aug 2017.

[60] R. Maaloul, R. Taktak, L. Chaari, and B. Cousin, "Energy-aware routing in carrier-grade ethernet using sdn approach," *IEEE Transactions on Green Communications and Networking*, 2018.

[61] J. Ba, Y. Wang, X. Zhong, S. Feng, X. Qiu, and S. Guo, "An sdn energy saving method based on topology switch and rerouting," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, April 2018.

[62] B. Puype *et al.*, "Multilayer Traffic Engineering for Energy Efficiency," *Photonic Netw. Commun.*, vol. 21, pp. 127–140, Apr. 2011.

[63] K. Bilal, S. U. Khan, S. A. Madani, K. Hayat, M. I. Khan, N. Min-Allah, J. Kolodziej, L. Wang, S. Zeadally, and D. Chen, "A survey on green communications using adaptive link rate," *Cluster Computing*, vol. 16, no. 3, pp. 575–589, 2013.

[64] G. Xu, B. Dai, B. Huang, and J. Yang, "Restorable energy aware routing with backup sharing in software defined networks," *Journal of Communications*, vol. 10, 2015.

[65] X. Chang, "Network simulations with opnet," in *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pp. 307–314, ACM, 1999.

[66] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[67] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 618–624, IEEE, 2013.

[68] Y. Carlinet and N. Perrot, "Energy-efficient load balancing in a sdn-based data-center network," in *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pp. 138–143, Sep. 2016.

[69] P. Habibi, M. Mokhtari, and M. Sabaei, "Qrve: Qos-aware routing and energy-efficient vm placement for software-defined datacenter networks," in *Telecommunications (IST), 2016 8th International Symposium on*, pp. 533–539, IEEE, 2016.

[70] M. Tsygankov and C. Chen, "Network aware vm load balancing in cloud data centers using sdn," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–6, June 2017.

[71] Y. Hu, T. Luo, N. C. Beaulieu, and W. Wang, "An initial load-based green software defined network," *Applied Sciences*, vol. 7, no. 5, p. 459, 2017.

[72] Q. Liao and Z. Wang, "Energy consumption optimization scheme of cloud data center based on sdn," *Procedia Computer Science*, vol. 131, pp. 1318 – 1327, 2018. Recent Advancement in Information and Communication Technology:.

[73] J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, pp. 76–89, April 2017.

[74] A. Akbar Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, "Nature-inspired meta-heuristic algorithms for solving the load balancing prob-

lem in the software-defined network," *International Journal of Communication Systems*, p. e3875, 2019.

[75] B. Kantarci, L. Foschini, A. Corradi, and H. T. Mouftah, "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *2012 IEEE Globecom Workshops*, pp. 708–713, IEEE, 2012.

[76] L. Zhou, L. N. Bhuyan, and K. K. Ramakrishnan, "Dream: Distributed energy-aware traffic management for data center networks," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy '19, (New York, NY, USA), pp. 273–284, ACM, 2019.

[77] G. Grid, "The green grid." `https://www.thegreengrid.org`, 2017. Accessed: 2017-12-30.

[78] SPEC, "Standard Performafnce Evaluation Corporation, url = `https://www.spec.org`, note = Accessed: 2019-01-20," 2019.

[79] G. Xu, B. Dai, B. Huang, J. Yang, and S. Wen, "Bandwidth-aware energy efficient flow scheduling with sdn in data center networks," in *Future Generation Computer Systems*, vol. 68, pp. 163–174, Elsevier, 2017.

[80] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[81] R. Garroppo, G. Nencioni, L. Tavanti, and M. G. Scutella, "Does traffic consolidation always lead to network energy savings?," *IEEE Communications Letters*, vol. 17, no. 9, pp. 1852–1855, 2013.

[82] R. Bolla, R. Bruschi, F. Davoli, and C. Lombardo, "Fine-grained energy-efficient consolidation in sdn networks and devices," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 132–145, 2015.

[83] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 18–33, 2015.

[84] M. Levy and D. Raviv, "A novel framework for data center metrics using a multidimensional approach," in *15th LACCEI International Multi-Conference for Engineering, Education, and Technology: Global Partnerships for Development and Engineering Education*, 2017.

[85] D. Vemula, B. Setz, G. S. V. Rao, G. Gangadharan, and M. Aiello, "Metrics for sustainable data centers," *IEEE Transactions on Sustainable Computing, 2017*, 2017.

[86] M. Levy and D. Raviv, "An overview of data center metrics and a novel approach for a new family of metrics," *Advances in Science, Technology and Engineering Systems J*, vol. 3, no. 2, pp. 238–251, 2018.

[87] E. Jaureguialzo, "Pue: The green grid metric for evaluating the energy efficiency in dc (data center). measurement method using the power demand," in *Telecommunications Energy Conference (INTELEC), 2011 IEEE 33rd International*, pp. 1–8, IEEE, 2011.

[88] D. Azevedo and M. Patterson, "Carbon usage effectiveness (cue): a green grid data center sustainability metric,2010," *The Green Grid*, 2010.

[89] D. Azevedo, S. C. Belady, and J. Pouchet, "Water usage effectiveness (wue™): A green grid data center sustainability metric,2011," *The Green Grid*, 2011.

[90] L. R. Prete, A. A. Shinoda, C. M. Schweitzer, and R. L. S. de Oliveira, "Simulation in an sdn network scenario using the pox controller," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1–6, IEEE, 2014.

[91] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007.

[92] F. Kaup, S. Melnikowitsch, and D. Hausheer, "Measuring and modeling the power consumption of openflow switches," in *Network and Service Management (CNSM), 2014 10th International Conference on*, pp. 181–186, IEEE, 2014.

[93] OvSwitch, "Production quality, multilayer open virtual switch," 2018. Accessed: 2018-12-26.

[94] Z. FX, "Zodiac FX OpenFlow Switch Hardware-Northbound Networks." `https://northboundnetworks.com/products/zodiac-fx`, 2018. Accessed: 2018-12-26.

[95] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[96] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[97] L. Yanjun, L. Xiaobo, and Y. Osamu, "Traffic engineering framework with machine learning based meta-layer in software-defined networks," in *Network Infrastructure and Digital Content (IC-NIDC), 2014 4th IEEE International Conference on*, pp. 121–125, IEEE, 2014.

[98] K. Sideris, R. Nejabati, and D. Simeonidou, "Seer: Empowering software defined networking with data analytics," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pp. 181–188, Dec 2016.

[99] A. Azzouni, R. Boutaba, and G. Pujolle, "Neuroute: Predictive dynamic routing for software-defined networks," *CoRR*, vol. abs/1709.06002, 2017.

[100] F. Francois and E. Gelenbe, "Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing," in *IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 283–288, Sep. 2016.

[101] R. Maaloul, L. Chaari, and B. Cousin, "Energy saving in carrier-grade networks: A survey," *Computer Standards & Interfaces*, 2017.

[102] P. Pudil and J. Hovovicova, "Novel methods for subset selection with respect to problem knowledge," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 2, pp. 66–74, 1998.

[103] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[104] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[105] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Dimension*, vol. 9, no. 7, 2006.

[106] I. K. Fodor, "A survey of dimension reduction techniques," tech. rep., Lawrence Livermore National Lab., CA (US), 2002.

[107] M. Demirci, "A survey of machine learning applications for energy-efficient resource management in cloud computing environments," in *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pp. 1185–1190, IEEE, 2015.

[108] M. Latah and L. Toker, "Artificial intelligence enabled software-defined networking: a comprehensive overview," *IET Networks*, vol. 8, no. 2, pp. 79–99, 2018.

[109] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 670–674, May 2017.

[110] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080*, 2017.

[111] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: a reinforcement learning approach," in *Services Computing (SCC), 2016 IEEE International Conference on*, pp. 25–33, IEEE, 2016.

[112] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*, pp. 167–172, IEEE, 2016.

[113] J. Ashraf and S. Latif, "Handling intrusion and ddos attacks in software defined networks using machine learning techniques," in *Software Engineering Conference (NSEC), 2014 National*, pp. 55–60, IEEE, 2014.

[114] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[115] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*, pp. 1137–1143, Morgan Kaufmann Publishers Inc., 1995.

[116] R. S. Sutton, *Introduction to reinforcement learning*, vol. 135. Cambridge: MIT press., 1999.

[117] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, pp. 671–678, 1994.

[118] Y. V. Potdar and M. S. Phatak, "Reinforcement learning algorithms in routing:" q_routing" implementation and analysis in ns-2," 2006.

[119] A. Esfahani and M. Analoui, "Widest k-shortest paths q-routing: A new qos routing algorithm in telecommunication networks," in *2008 International Conference on Computer Science and Software Engineering*, vol. 4, pp. 1032–1035, Dec 2008.

[120] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains," tech. rep., Carnegie-Millon University Pittsburgy PA school of Computer Science, 1992.

[121] N. Sabharwal, *Apache CloudStack cloud computing*. Packt Publishing Ltd, 2013.

[122] A. Shrivastwa, S. Sarat, K. Jackson, C. Bunch, E. Sigler, and T. Campbell, *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, 2016.

[123] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment.," *IEEE network*, vol. 29, no. 2, pp. 56–61, 2015.

[124] C. Ghribi, *Energy efficient resource allocation in cloud computing environments*. PhD thesis, UPMC - University Pierre and Marie CURIE, 2014.

[125] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[126] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1385–1400, 2018.

[127] A. Wang, Z. Zha, Y. Guo, and S. Chen, "Software-defined networking enhanced edge computing: A network-centric survey," *Proceedings of the IEEE*, vol. 107, pp. 1500–1519, Aug 2019.

[128] R. Amin, M. Reisslein, and N. Shah, "Hybrid sdn networks: A survey of existing approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[129] J. Bhatia, Y. Modi, S. Tanwar, and M. Bhavsar, "Software defined vehicular networks: A comprehensive review," *International Journal of Communication Systems*, vol. 32, no. 12, p. e4005, 2019.