Network-aware energy saving techniques in cloud data centers

Motassem Al-Tarazi
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd
Part of the Computer Sciences Commons

Recommended Citation
Al-Tarazi, Motassem, "Network-aware energy saving techniques in cloud data centers" (2019). Graduate Theses and Dissertations. 16960.
https://lib.dr.iastate.edu/etd/16960

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Network-aware energy saving techniques in cloud data centers

by

Motassem Al-Tarazi

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
J. Morris Chang, Co-major Professor
Ying Cai, Co-major Professor
Wensheng Zhang
Lu Ruan
Nathan Neihart

The student author, whose presentation of the scholarship herein was approved by the program of
study committee, is solely responsible for the content of this dissertation. The Graduate College
will ensure this dissertation is globally accessible and will not permit alterations after a degree is
conferred.

Iowa State University
Ames, Iowa
2019

Copyright © Motassem Al-Tarazi, 2019. All rights reserved.
DEDICATED

I would like to dedicate this thesis to my Mom Eman Jaran and to my Dad Yaser Tarazi, without your love and support I would not have been able to complete this work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2. PERFORMANCE-AWARE ENERGY SAVING FOR DATA CENTER NETWORKS</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Abstract</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Related works</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Problem Formulation</td>
<td>10</td>
</tr>
<tr>
<td>2.5 System Model</td>
<td>14</td>
</tr>
<tr>
<td>2.5.1 Network Topologies</td>
<td>14</td>
</tr>
<tr>
<td>2.5.2 Traffic Model</td>
<td>15</td>
</tr>
<tr>
<td>2.6 Heuristic Approach</td>
<td>15</td>
</tr>
<tr>
<td>2.6.1 Performance Bound Analysis</td>
<td>19</td>
</tr>
<tr>
<td>2.7 Load Balancing</td>
<td>20</td>
</tr>
<tr>
<td>2.8 Performance Evaluation</td>
<td>21</td>
</tr>
<tr>
<td>2.8.1 Simulation Setup</td>
<td>22</td>
</tr>
<tr>
<td>2.8.2 Network Energy Consumption</td>
<td>24</td>
</tr>
<tr>
<td>2.8.3 Network Throughput</td>
<td>25</td>
</tr>
<tr>
<td>2.8.4 Average End-to-End Delay</td>
<td>26</td>
</tr>
<tr>
<td>2.8.5 Ratio of Dropped Packets</td>
<td>27</td>
</tr>
<tr>
<td>2.8.6 Average Imbalance Score</td>
<td>29</td>
</tr>
<tr>
<td>2.8.7 Real Packet Traces</td>
<td>31</td>
</tr>
<tr>
<td>2.8.8 More Comparisons</td>
<td>32</td>
</tr>
<tr>
<td>2.8.9 CPLEX versus Heuristic Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>2.9 Conclusion</td>
<td>35</td>
</tr>
<tr>
<td>CHAPTER 3. NETWORK-AWARE ENERGY SAVING MULTI-OBJECTIVE OPTIMIZATION IN VIRTUALIZED DATA CENTERS</td>
<td>37</td>
</tr>
<tr>
<td>3.1 Abstract</td>
<td>37</td>
</tr>
<tr>
<td>3.2 Introduction</td>
<td>37</td>
</tr>
</tbody>
</table>
### 3.3 Related works ................................................................. 41
### 3.4 System Model ............................................................... 42
### 3.5 Problem Formulation ................................................... 43
### 3.6 Heuristic Approach ..................................................... 49
### 3.7 Evaluation and Experimental Evaluation .......................... 52
#### 3.7.1 Testbed Setup ....................................................... 53
#### 3.7.2 Testbed Results ..................................................... 54
#### 3.7.3 CPLEX versus Heuristic Algorithm Results .................. 59
### 3.8 Conclusion and Future Works ....................................... 60

#### CHAPTER 4. PREDICTION-BASED JOINT ENERGY SAVING OPTIMIZATION FOR VIRTUALIZED DATA CENTERS ......................................................... 62
### 4.1 Abstract ................................................................. 62
### 4.2 Introduction ............................................................. 62
### 4.3 Related works .......................................................... 65
### 4.4 System Model ........................................................... 66
### 4.5 Resource Measurement and Prediction .............................. 67
### 4.6 Problem Formulation .................................................. 68
### 4.7 Heuristic Algorithm ................................................... 73
### 4.8 Performance Evaluation ............................................ 77
#### 4.8.1 Simulation Setup .................................................. 77
#### 4.8.2 Simulation results ............................................... 79
### 4.9 Conclusion ............................................................. 81

#### CHAPTER 5. CONCLUSIONS AND FUTURE WORKS .................. 83

#### BIBLIOGRAPHY ............................................................ 86
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Definition of important symbols</td>
<td>11</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>System parameters</td>
<td>22</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>Power rates of various commodity switches in Watts</td>
<td>23</td>
</tr>
<tr>
<td>Table 2.4</td>
<td>Network and tasks parameters</td>
<td>23</td>
</tr>
<tr>
<td>Table 2.5</td>
<td>Average imbalance scores</td>
<td>29</td>
</tr>
<tr>
<td>Table 2.6</td>
<td>Traces results with Fat tree ($k = 6$)</td>
<td>30</td>
</tr>
<tr>
<td>Table 2.7</td>
<td>Traces results with Three tier</td>
<td>31</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Definition of Important symbols</td>
<td>45</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Comparison of the number virtual machines migrations</td>
<td>58</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Comparison of computational time between the proposed heuristic algorithm and CPLEX (in seconds)</td>
<td>60</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Definition of important symbols</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Data center network topologies: (a) Three-Tier. (b) Fat Tree with K = 4.</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Comparison of links loads: (a) CDF at Core Level. (b) CDF at Aggregation Level.</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Network energy consumption at different data center loads. (a) Fat tree. (b) Three-Tier.</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Network throughput at different data center loads. (a) Fat tree. (b) Three-Tier.</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Network end-to-end delay at different data center loads. (a) Fat tree. (b) Three-Tier.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Ratio of dropped packets at different data center loads. (a) Fat tree. (b) Three-Tier.</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Comparison of power consumption between heuristic algorithm and CPLEX.</td>
<td>34</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>Comparison of computation time between heuristic algorithm and CPLEX.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Proposed system framework.</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Testbed network topology.</td>
<td>53</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Network bandwidth consumption using DRS.</td>
<td>55</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Network bandwidth consumption using heuristic algorithm.</td>
<td>55</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Testbed energy consumption.</td>
<td>56</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Testbed computational running time.</td>
<td>57</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Comparison of energy consumption between the proposed heuristic algorithm and CPLEX.</td>
<td>58</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Proposed system framework.</td>
<td>67</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.2</td>
<td>Fat tree network topology with $k = 4$.</td>
<td>78</td>
</tr>
<tr>
<td>4.3</td>
<td>Wikipedia one day traces: Page view requests.</td>
<td>78</td>
</tr>
<tr>
<td>4.4</td>
<td>Wikipedia one day traces: Response size in bytes.</td>
<td>79</td>
</tr>
<tr>
<td>4.5</td>
<td>Servers energy consumption.</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>Switches energy consumption.</td>
<td>80</td>
</tr>
<tr>
<td>4.7</td>
<td>Total energy consumption.</td>
<td>81</td>
</tr>
<tr>
<td>4.8</td>
<td>Average response time.</td>
<td>81</td>
</tr>
</tbody>
</table>
I would like to take this opportunity to express my thanks to my wife Ala’a Abu Basha and my daughter Celia for their selfless love, encouragement and support. Also, I would like to thank my family and friends who provided me with help, support, and advice throughout this journey. My adviser Professor Morris Chang, thank you for your guidance, patience and support throughout this research. Your insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my co-adviser Dr. Ying Cai and the committee members for their efforts and contributions to this work: Dr. Wensheng Zhang, Dr. Nathan Neihart and Dr. Lu Ruan.
ABSTRACT

In recent years, the growth and popularity of cloud computing services is leading toward the rise of large-scale data centers. Data centers are one of the most energy consumed categories in the world. How to efficiently save data center energy while maintaining its performance is one of the most important research issues in the field of cloud computing.

In this thesis, we try to tackle the following research problems: 1. how to achieve considerable amount of energy saving in cloud data centers; 2. how to maintain data center performance, 3. how to provide a practical and scalable solutions that can be implemented in modern enterprise data centers. We addressed those research problems by proposing and analyzing different for saving energy in data centers: The first model focuses on saving data center network energy while preserving network performance. The idea is to use route consolidation to switch traffic to a small number of network devices and turn off unused devices and links. To maintain network performance, safety thresholds for links utilization and valiant load balancing on active switches are used.

The second model discusses the energy-saving problem for the server side of the data center. The model uses dynamic placement and live migration of virtual machines to save energy while taking into account the current status of the network. The model migrates virtual machines to a subset of servers and put unused servers into standby mode. At all times, the resource requirements for all virtual machines are maintained and the overhead introduced to the network by live migration is minimized.

The third model combines server and network sides to maximize energy saving while preserving network performance. The model takes advantage of network traffic and virtual machines consolidation techniques to focus the workloads to a subset of devices and puts others to off or standby mode. The model is part of a framework that monitors the state of the data center by collecting and predicting run time utilization data for servers’ resources (CPU, memory, network, and disk).
and network traffic. It uses them as an input. The model will provide a new virtual machines placement and flow routing matrix that assure maximum data center energy saving while maintaining performance. Migration commands will take place to adjust the placement of the virtual machines based on the solution. Finally, unused servers are moved to standby mode and switches are turned off.
CHAPTER 1. INTRODUCTION

In general, energy saving options in virtualized data centers can be categorized into the following: First, proposing new topological designs that use a small number of links and devices that produce a similar performance to the original. Although these new topologies save energy efficiently, the primary drawback of these new topologies is that they cannot be applied to existing data centers as they require specific hardware and software capabilities. Second, providing local server level energy saving by using Dynamic Voltage and Frequency Scaling (DVFS). The amount of energy saved by this category is small compared to other categories. Moreover, servers may face performance degradation if a false prediction occurs. Third, introducing optimization problems for current Data Center Networks (DCNs) and propose different techniques and heuristics to solve them. Fourth, adopting VM placement and migration problem, where the VMs are migrated to a subset of the servers and turn the rest into sleeping mode. Finally, combining the network energy-saving problem with the server energy saving problem to maximize the overall energy saving. The last three categories share the same principle, moving workloads or traffic to a subset of devices and put unused devices to sleep or standby mode. The cost of applying such techniques is not high compared to adopting new topological design, and the energy that can be saved is high.

In fact, previous studies showed that the best way to save energy is via consolidation techniques. The idea is to consolidate traffic flows and virtual machines into subset of switches and servers and put unused servers to standby mode and turning off unused switches. In this thesis, we present models that follow the third, fourth and fifth categories to save network and servers energy with no or minimal effect on the data center network.

In chapter 2, an optimization model to save the energy of the data center network (DCN) with minimum or no effect on the network performance is proposed. The problem was formulated as a mixed integer linear program (MILP) with minimizing the consumed energy as the main
objective. The problem constrained by network performance constraints such as: maximum link utilization and marginal thresholds. For practical implementation to large data centers, a heuristic algorithm is proposed. The algorithm uses switches grouping and links consolidation to switch the traffic to a small number of network devices and turn-off unused switches and links. Meanwhile, safety threshold for links utilization is set to always have the ability to accept incoming packets in case of traffic surges. Furthermore, the algorithm uses Valiant Load Balancing (VLB) mechanism on active switches to spread the loads among the active switches. The algorithm was evaluated using GreenCloud simulator using synthetic and real traffic traces against benchmark algorithms. The results show that the propose algorithm can save up to 35% of the network energy with three-tier topology and up to 45% of the network energy with fat tree topology while maintain network performance. To evaluate the load balancing mechanism used in the proposed algorithm, the average imbalance score for both links and switches was proposed. The imbalance score of switches/links is the standard deviation of the average switch/link utilization for all switches/links in a switching level. The proposed algorithm improves the imbalance scores for both links and switches by more than 50% and 60%, respectively. Lastly, the proposed algorithm solution was evaluated against the optimal solution obtained by CPLEX. The results show that the power consumption gap between the solution provided by the proposed algorithm and the optimal solution provided by CPLEX is less than 4%. In comparison, the computational time for CPLEX is very high compared to the proposed algorithm.

In Chapter 3, a multi-objective optimization is proposed with the aim of saving data center energy and, at the same time, minimizing the overhead introduced to the network by live migration. The problem was formulated as a weighted-sum multi-objective optimization with minimizing the consumed energy and the time to migrate virtual machines are the main objectives. A two-stage heuristic algorithm is presented for practical implementation. The first stage tries to find an initial feasible solution to satisfy all virtual machines resource requirements. Once found, the second stage is invoked to find a better solution for saving energy while minimizing the overhead resulting from adjusting the placements of virtual machines to the network. To achieve this solution, the heuristic
algorithm consolidates virtual machines to a small subset of servers that satisfy their requirements and put unused servers to standby mode. The heuristic searches for the best routes to consolidate virtual machines with minimum effect on the network links. The heuristic is evaluated using a real testbed data center. Hadoop 2.7.3 multi-node cluster is deployed on the testbed to mimic real data centers environment, while the testbed is stressed using different types of MapReduce jobs from Hibench. The experiments show that the proposed framework can save energy up to 30% while achieving better performance with minimum effect on the data center network.

In chapter 4, we combine the server and network energy consumption in virtualized data centers while maintaining performance. The problem was formulated as a joint MILP to minimize the energy consumed by the servers and the network. The joint optimization is part of a framework that monitors the state of the data center by collecting and predicting run time utilization data for servers’ resources (CPU, memory, network, and disk) and network traffic. It uses them as an input for the joint optimization. The joint objective optimization will provide a new virtual machines placement and flow routing matrix that assure maximum data center energy saving while maintaining performance. Live migration commands will take place to adjust the placement of the virtual machines into their designated destinations based on the optimization solution. Finally, unused servers are moved to standby mode and switches are turned off. As the proposed formulation is NP-hard, a two stages heuristic algorithm is proposed. The first stage starts at the servers’ side, the virtual machines initial placed using First Fit Decreasing. After that, the heuristic uses the resource predictions to solve resource utilization violations and save more energy. The second stage includes the use of an abstract performance aware flow routing and consolidation. The proposed heuristic was evaluated using CloudsimSDN simulator using traces collected from Wikipedia page view statistic. The results show that the proposed algorithm can save significant amount of energy in both servers and network sides while maintaining performance represented by average response time.
CHAPTER 2. PERFORMANCE-AWARE ENERGY SAVING FOR DATA CENTER NETWORKS

A paper accepted by *IEEE Transactions on Network and Management Service*

Motassem Al-Tarazi and J. Morris Chang

2.1 Abstract

Today’s data center networks (DCNs) tend to have tens to hundreds of thousands of servers to provide massive and sophisticated services. The architectural design of DCNs usually over-provisioned for peaks workloads and fault-tolerance. Statistically, DCNs remain highly under-utilized with typical utilization of around 30%. Network over-provisioning and under-utilization can be exploited for energy-saving. Most research efforts on data center network energy saving focus on how to save maximum energy with little or no consideration to the performance of the residual network. Thus, the DCN performance degraded and the network left vulnerable to sudden traffic surges. In this paper, we have studied energy-saving problem in DCNs while preserving network performance. The problem was formulated as MILP that is solvable by CPLEX to minimize the energy consumed by DCN, meanwhile, safety threshold constraints for links utilization are met. To overcome CPLEX high computational time, a heuristic algorithm to provide practical and efficient solution for the MILP is introduced. The heuristic algorithm uses switches grouping and links consolidation to switch the traffic to a small number of network devices and turn-off unused switches and links. Valiant load-balancing is used to distribute the loads over active links. Simulation experiments using synthetic and real packet traces were conducted to validate the heuristic in terms of energy consumption and network performance. The results show that the heuristic can save up to 45% of the network energy and improves the average imbalance-scores for links and switches by more than 50% with minimal effect on network performance.
2.2 Introduction

Currently, data center networks (DCNs) tend to have tens to hundreds of thousands of servers to provide massive and sophisticated services, such as web searching, cloud storage, online social services, and scientific computing. As data centers become more popular, the importance of power consumption issues is increased due to the high number of powered devices [1]. EPA reported that the total electricity used by data centers in 2010 was about 1.3% of all electricity used in the world [2] and it is expected to reach 8% by 2020 [3].

Extensive research has been done on the energy saving techniques for the server side of the data centers, while the problem for the network side is still a substantial issue. Today’s DCNs designed to accommodate peak loads in most reliable way without taking energy saving into consideration. Data center networks are built with many redundant links and heavily over-provisioned link bandwidth to handle link failures and traffic bursts. Although current data centers design increases reliability, it also decreases energy efficiency since all network devices are powered-on all the time with minimal link utilization. Statistics showed that most of the network devices are under-utilized, where the typical utilization of a DCN is only 30% [4]. DCNs’ over-provisioning and under-utilization can be exploited for energy saving research.

Existing research proposed many techniques to overcome the energy saving problem. A stream of research [5; 6; 7] proposed energy efficient network topologies. Although these topologies reduce energy efficiently, for example, the optical-based topologies Proteus [8] and Petbit [9] were reported to save up to 75% of the data center power consumption, applying them to existing DCNs is expensive and require hardware modification. Another stream of research focused on traffic engineering and route consolidation as in [10; 11; 12; 13]. The main idea of this stream is to turn the network load to a minimal subset of network devices. Then it puts unused devices to sleep mode or shut them down to minimize the overall network power consumption. Using traffic engineering and route consolidation, there will always be a trade-off between energy saving and performance.

Few studies discussed this trade-off. Zhang et al. [14] proposed a traffic engineering technique to maximize the number of links that can be shut down under some network performance constraints,
such as link utilization and packet delay. In their study no techniques were specified to handle traffic bursts. Shang et al. [15] proposed an energy aware routing, the idea is to use a few devices to satisfy the network demand with little or no degradation in the overall performance represented by the throughput of the original network. Initially, they compute the network throughput by routing all network devices; start to remove switches until the throughput decreases to a predefined threshold. Finally, switches not involved in the final routing are either powered off or put into sleep mode. This technique suffers from inefficient computational running time as it takes long time to calculate a near optimal solution.

In this paper, we studied the problem of saving data center network energy while maintaining network performance against traffic surges. The problem is formulated as a mix integer linear problem (MILP) to minimize the total network energy as a main objective. Moreover, the problem was constrained by network performance requirements, such as maximum link utilization with safety margin threshold. In general, MILPs are NP-hard problems, thus, the computational time to solve MILP increases exponentially with the size of the problem. For example, the time to find the optimal solution for the MILP using a data center network with 54000 servers is more than 4 hours. Therefore, solving the energy saving problem for large data centers is impractical.

For practical implementation to large data center networks, we argue that setting margin threshold alone, as in existing methods such as [16], is not enough for saving energy and maintaining network performance from traffic surges. So, a light-weight heuristic algorithm that combines setting-up safety margin threshold and load balancing technique together is presented to save energy and maintain network performance to handle traffic surges.

The heuristic algorithm starts by setting up predefined safety thresholds on each link capacity. Then, it continuously monitors the utilization of network links and balances the loads on active links using Valiant Load Balancing (VLB) mechanism [17]. A decision to turn on new switches or links can be taken if these thresholds are exceeded. Using this algorithm, the safety margins and the load balancing mechanism allow the network to handle traffic surges, while maintaining its performance. On the other hand, switches grouping and links consolidation will also take place if
the loads on the networks switches and links are under-utilized. This will allow turning off some active ports and switches to lower network power consumption.

To validate the effectiveness of the algorithm, extensive simulations conducted on data centers with classical three-tier and fat tree [18] topologies. The proposed algorithm was evaluated against data centers without any energy saving mechanism, data centers with greedy bin-packing energy saving mechanism, and Global First Fit energy saving mechanism in terms of energy saving, average end to end delay, throughput and drop packets at various data center loads. The results showed that the proposed algorithm can save up to 35% with three-tier topology and up to 45% with the folded clos fat tree topology with minor effect on network performance. To evaluate the load balancing mechanism used in the proposed algorithm, the imbalance scores for both links and switches are compared against the same proposed algorithm without any load balancing mechanisms. The imbalance score of switches/links is the standard deviation of the average switch/link utilization for all switches/links in a switching level. The proposed algorithm improves the imbalance scores for both links and switches by more than 50% and 60%, respectively. In addition, the proposed algorithm solution was evaluated against the optimal solution obtained by CPLEX [19] in terms of power consumption and computational running time. The results show that the power consumption gap between the solution provided by the proposed algorithm and the optimal solution provided by CPLEX is less than 4%. In comparison, the computational time for CPLEX is very high compared to the proposed algorithm.

The list of contributions in this paper is as follows:

- We propose a technique to save data center energy while preserving network performance from traffic surges. The problem was formulated as a mixed integer linear program. We identify that setting up link utilization threshold alone is not enough to preserve network performance as shown in our evaluation. We proposed that in addition of setting link utilization threshold a load balancing technique should be applied to preserve network performance and handle sudden traffic surges.
For large scale data centers, we design a heuristic algorithm that sets safety thresholds on link capacities and uses valiant load balancing technique on active links. The proposed heuristic is abstract and can be applied to any switch-centric topology in similar fashion.

We implement the proposed heuristic algorithm using GreenCloud simulator and compared to the base case, Greedy bin-packing, Global first fit, and the proposed heuristic without load balancing. Both synthetic and real traces demonstrate that the heuristic algorithm saves considerable amount of energy with minimum effect on the DCN.

We propose the Average Imbalance Score metric for both switches and links to evaluate the performance of the load balancing mechanism. Using this metric, we show that the heuristic algorithm improves the imbalance score for links and switches by more than 50%.

The rest of the paper is organized as follows. Section 2.3 reviews previous related works. Section 2.4 formulates the power saving problem. Section 2.5 presents the system model used. Section 2.6 proposes the heuristic algorithm. Section 2.7 discusses how the heuristic algorithms achieves a desirable amount of load balancing. Section 2.8 presents the simulation experiments and discusses the results and finally Section 2.9 concludes the paper.

### 2.3 Related works

Many approaches have been proposed to deal with the data center network energy saving problem. A number of researchers proposed designs of new topological structures that provide energy conservation while preserving performance. Examples may include flatted butterfly [20], Pcube [5], Small-World [6], NovaCube [7], 3D Torus based CamCube [21], and Proteus [8]. The primary drawback of these new topologies is that they cannot be applied to existing data centers as they require specific hardware and software capabilities.

On the other hand, some researchers found optimization problems for current DCNs and propose different techniques and heuristics to solve them. ElasticTree [16] proposed a power manager that adjusts the active switches and links to satisfy dynamic traffic loads. The authors in [16] proposed
setting safety margins to provide performance insurance by delaying the point at which packets start to drop and latency starts to degrade. Carpo [22] introduced a correlation-aware power optimization algorithm, it dynamically consolidates traffic loads into a minimal set of switches and links and shut down unused devices. REsPoNse [23] discussed the trade-off between optimal energy saving and scalability. It identifies a few critical routes offline, installs them to routing tables, then runs an online simple scalable traffic engineering to activate and deactivate network devices. GreenTE [14] proposed a power-aware traffic engineering model. They try to maximize the number of links that will be shut down under certain constraints such as maximum link utilization and packet delay. Wang et al. proposed a rate adaptation approach for future data center networks to solve the oscillation brought by traffic engineering approaches [24].

In [10], the authors introduced a combination between energy-aware routing and preemptive flow scheduling to maximize energy efficiency. [15] Introduced a model that uses few network devices as possible to provide routing services with little or no sacrifice of the network throughput. They compute the network throughput according to routing overall network devices; start to remove switches until the throughput decreases to a predefined threshold, and finally switches not involved in the final routing are either powered off or put into sleep mode. [11] proposed PowerNets, a power optimization framework that minimizes DCN, server, and cooling power together. For more energy saving, a workload correlation analysis is done during the server and traffic consolidation processes. The authors in [25] present PowerFCT, an energy saving scheme that combines flow consolidation and DCNs’ switch components power throttling. They also consider flow completion time of delay sensitive flows to preserve network performance. [12] designed a power efficient network system that is based on an artificial intelligence abstraction model called blocking island. The idea is to produce a set of different B-blocking islands and blocking island hierarchy tree (BIH) based on the available bandwidth. For each traffic demand, bandwidth allocation mechanism and power-aware routing algorithm are applied on BIH to compute and allocate the best routing path. After having a set of routes that satisfy all the demands, backup routes will be added for fault tolerance. Finally, the system turns off or puts to sleep the switches, line cards, or links not in the solution set.
A distributed flow consolidation framework with correlation analysis and delay constraints presented by [26]. They present two distributed heuristics that provide different trade-offs between scalability, power saving, and network performance. A flow consolidation that considers the flow completion time (FCT) introduced by [13]. It is designed based on control theory to dynamically control the FCT of delay of delay-sensitive traffic flows. Merge network [27] considers minimizing the power consumed by a switch. It tries to consolidate links loads to a smaller subset of links within the same switch, then turning the unused links to low power mode. This approach focuses on reducing energy within switches, which tend to have less energy-saving compared to traffic engineering approaches which tries to merge traffic at a subset of the network. Most of these techniques focus on the optimization without setting load balancing and safeguards policies to maintain the performance of the network.

Another stream of research tried to combine the network energy-saving problem with the server energy saving problem to maximize the overall energy saving as in [28; 29] or to combine it with VM placement problem as in [30; 31; 32; 33]. This combination will add extra load to the network due to VM migration overhead. Although this combined mechanism will provide extra energy saving, the network performance will suffer due to route consolidation and VM migration.

2.4 Problem Formulation

Consider a data center network \( G = (V, E) \) where \( V \) is the set of nodes and \( E \) is the set of links. A port link can be turned-off if there is no traffic on the link and a switch can be turned-off if all its ports are turned-off.

Let \( S \) be the set of all switches in the network where \( S \subseteq V \). The power consumed by a single switch \( s \in S \) consists of fixed power \( \epsilon_{Fixed} \), which consumed by components like (chassis, fans, etc), and ports power \( \epsilon_{Port} \). The power saving gained from turning off a single port is \( \epsilon_{Port} \), and from turning off an entire switch is \( \epsilon_{Fixed} + \sum_{i \in N_i} \epsilon_{Port} \). We use \( On(i) \) and \( On(i, j) \) as decision variables to denote that switch \( i \) and link \( (i, j) \) are active or not.
Table 2.1 Definition of important symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>Set of all switches</td>
</tr>
<tr>
<td>$S_{C}, S_{Agg}, S_{Acc}$</td>
<td>Sets of core, aggregation, access switches respectively</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of all ports</td>
</tr>
<tr>
<td>$E$</td>
<td>Set of all links</td>
</tr>
<tr>
<td>$D$</td>
<td>Set of all traffic Demands</td>
</tr>
<tr>
<td>$N_i$</td>
<td>Ports in switch $i$</td>
</tr>
<tr>
<td>$i, j$</td>
<td>A link connects two nodes $i$ and $j$</td>
</tr>
<tr>
<td>$e_{Port}$</td>
<td>Energy consumed by a port</td>
</tr>
<tr>
<td>$e_{Fixed}$</td>
<td>Fixed energy consumed by a switch</td>
</tr>
<tr>
<td>$f_{i,j}^t$</td>
<td>Traffic flow $t$ through link $i, j$</td>
</tr>
<tr>
<td>$C$</td>
<td>Capacity Matrix for all links</td>
</tr>
<tr>
<td>$On(i, j)$</td>
<td>A 0,1 Decision variable indicates if the link is on or off</td>
</tr>
<tr>
<td>$On(s)$</td>
<td>A 0,1 Decision variable indicates if the switch is on or off</td>
</tr>
<tr>
<td>$P(s)$</td>
<td>The total power consumed by switch $s$</td>
</tr>
<tr>
<td>$U_{upper}$</td>
<td>Upper link utilization threshold</td>
</tr>
<tr>
<td>$u_{i,j}$</td>
<td>Utilization of link $i, j$</td>
</tr>
</tbody>
</table>

Assume that the traffic demand matrix $D$ consists of a number of flows $\{f^0, f^1, \ldots, f^t\}$. Each flow $f^t$ will be passing through a number of links from source to destination that satisfies the flow load. $f_{i,j}^t$ represents the flow load of $t$ that is passing through link $(i, j)$. The traffic matrix $\tau$ is the summation of all flow loads passing through each link in the data center network $G$. Note that each link $(i, j) \in E$ has a bidirectional bandwidth capacity $C_{i,j} \in C$, where $C$ is the capacity matrix for all links in $E$. With the notations summarized in Table 2.1, we can formulate our problem as a mixed integer linear program that is solvable by CPLEX as the following: The MILP takes the data center network $G(V, E)$, the demand matrix $D$, the capacity matrix $C$, and the upper utilization...
threshold $U^{upper}$ as input. Equation 2.1 is the objective function. It minimizes the network power consumption function $P(x)$ for every switch. Thus, minimizing the total power consumed by the data center network.

$$\text{Minimize } \sum_{x \in S} P(x) \quad (2.1)$$

The constraints are divided into three categories: links constraints, switches constraints, and utilization constraints. Equations 2.2-2.5 present network links constraints. Equation 2.2 introduces the active link constraint [29]. It states that an active link connects two active switches or a switch and a server.

$$On(i, j) \leq On(i), On(i, j) \leq On(j) \forall i, j \in E, \forall i, j \in S \quad (2.2)$$

Equation 2.3 states the bidirectional link power constraint which means both directions of a link $(i, j)$ should have the same on/off power status. Likewise, equation 2.4 ensures that for every active link $On(i, j) = 1$, both directions have the same capacity limits $C_{i,j}$.

$$On(i, j) = On(j, i) \forall i, j \in E \quad (2.3)$$

$$On(i, j) \cdot C_{i,j} = On(i, j) \cdot C_{j,i} \forall i, j \in E \quad (2.4)$$
Equation 2.5 introduces the satisfiability constraint. It shows that the summation of all traffic flow loads $\sum_{t=0}^{n} f_t^l$ passing through link $(i,j)$ is always less than or equal to the capacity limit of that link $C_{i,j}$. Where $n$ is the number of all traffic flows.

$$\sum_{t=0}^{n} f_t^l \leq On(i,j) \cdot C_{i,j}, \forall i, j \in E \quad (2.5)$$

Equations 2.6-2.7 present network switch constraints. Equation 2.6 shows the active switch constraint. Let $N_i \in N$ be the set of ports in a switch and $|N_i|$ is the cardinality of $N_i$, then equation 2.6 ensures that a switch will be turned off only if all its ports are turned off.

$$|N_i| \cdot (1 - On(i)) \leq \sum_{j \in N_i} (1 - On(i,j)), \forall i, j \in E, \forall i \in S \quad (2.6)$$

Equation 2.7 calculates the power consumed by a switch. Which is the power consumed by its fixed components $\epsilon^{Fixed}$, such as chassis, fans, line cards, ... etc., in addition to the power consumed by each active port $\epsilon^{Port}$.

$$P(x) = \epsilon^{Fixed} \cdot On(x) + \sum_{n \in N_i} \epsilon^{Port} \cdot On(x,n) \quad (2.7)$$

Equations 2.8-2.9 present utilization constraints. Equation 2.8 calculates the link utilization $u$ for each link. Where link utilization is the summation of every traffic flow load passing link $(i,j)$ to the capacity of that link. Equation 2.9 ensures that the utilization of every link is always less than or equal to a predefined upper link utilization threshold $U^{upper}$ (in this paper $U^{upper} = 0.80$).

$$u_{i,j} = \frac{\sum_{t=0}^{n} f_t^l}{C_{i,j}}, \forall i, j \in E \quad (2.8)$$

$$u_{i,j} \leq U^{upper}, \forall i, j \in E \quad (2.9)$$

Equations 2.10-2.11 show the problem decision variables. $On(i,j)$ and $On(i)$ are binary decision variables indicate the power status for network links and switches, respectively. Since $On(i,j)$ and $On(i)$ are binary integers, the formulated problem is MILP.

$$On(i) \in 0, 1 \quad (2.10)$$
Since mixed integer linear programming is NP-hard, the proposed formulation is not practical for large data center networks. Thus, it can be used as a benchmark tool to evaluate practical heuristic approaches.

2.5 System Model

This section provides a brief background about network topologies and traffic model used.

2.5.1 Network Topologies

In this paper, we considered applying our technique to two of the most popular topologies in data center networks: Three-tier and Fat-tree topologies.

The three-tier is the mostly used topology in data center networks [34]. Three-tier topology consists of three switching layers; core or border routers, aggregation switches, and top-of-rack (ToR) access switches. Each ToR connects up to 48 servers placed in a rack with 1 Gbps links, while for redundancy issues; a ToR is connected to two aggregation switches. Furthermore, each aggregation switch is connected to core switches with multiple high speed 10 Gbps links [35]. Unfortunately, three-tier topology suffers from various issues such as: scalability, cost, energy consumption, cross-section bandwidth, and agility [36]. Figure 2.1(a) shows the three-tier topology.

The fat-tree topology in data center networks was proposed by [18] to deal with the issues of traditional data centers. Fat tree is a multi-rooted tree, where its links - unlike the traditional tree topologies - became larger in term of capacity as they move toward the roots. Fat-tree is one of the Clos technologies that has been adopted for Google data centers [37]. Figure 2.1(b) illustrates the fat-tree topology with $k = 4$.

In general, if $k$-port switches are used to construct a fat-tree, then $(k/2)^2$ core switches are needed to connect $k$ pods, each pod consists of $k/2$ access switches and $k/2$ aggregation switches. Within a pod, aggregation switches and access switches are connected with each other to form
a complete bipartite graph. Since each access switch connected to $k/2$ aggregate switches, each access switch is also connected to $k/2$ servers. So the number servers supported by a $k$-port fat tree are $k^3/4$.

### 2.5.2 Traffic Model

The traffic model used follows the ingress/egress traffic model. The ingress traffic represents accepted task requests by the data center, which travels from core switches down to its designated server. On the other hand, the egress traffic are the outputs of the tasks which originated at servers and traverse upward to core switches.

Let $n_s$ denote the number of servers/switches connected by one switch at any switching level on a switch-centric topology. Thus, the ingress/egress capacity limit of each switch is bounded by $n_s \cdot C_s$, where $C_s$ is the capacity of link $s$. Taking the upper link utilization threshold into account, the capacity limit will be bounded by $n_s \cdot C_s \cdot U^{upper}$. So, any valid traffic matrix $\tau \in D$ should satisfy the following constraint:

$$\sum_{s=0}^{n_s-1} \sum_{t=0}^{n-1} f_s^t \leq n_s \cdot C_s \cdot U^{upper}, \text{ where } s \in E$$  \hspace{1cm} (2.12)

### 2.6 Heuristic Approach

To overcome the exponential increase in CPLEX computation time, a heuristic algorithm solving the data center energy-saving problem was developed. In data center environment, traffic demands fluctuate frequently. For that reason, heuristic algorithm is preferred to solve our optimization model in real time.

Algorithm 1 illustrates the heuristic pseudocode, it takes similar inputs as in CPLEX. The output includes a set of active switches and ports that satisfies the traffic demands as well as the load balancing requirements. The heuristic algorithm devised to solve the problem under any switch-centric topology [38] in similar way.
Algorithm 1 Heuristic Algorithm

1: **Input:** $G(V, E), A, C, flow, U_{upper}$
2: **Output:** Set of active switches and ports $A$
3: $S_C \subseteq V; i \leftarrow 0; l \in N_s; N_s \in E$
4: if $A = \phi$ then
   5:     $A = \text{MST}( )$
6: $A' = A$
7: $A'' = \phi; A''' = S_C - A'$
8: while $A' \neq \phi$ do
9:     Randomly select $i \in A'$
10:     if $\exists l \in i$ is active then
11:         if $\text{LinkChecker}(flow, l, i, C_i,l)$ then
12:             Update($A,A', flow, i$); break;
13:     else
14:         if $\exists l \in i$ is inactive then
15:             $A'' = A'' + i$
16:     End if
17:     $A' = A' - i$
18: End While
19: if $A' = \phi$ then
20:     if $A'' \neq \phi$ then
21:         Randomly select $i \in A''$
22:         SET $l$ to active
23:     else
24:         Randomly select $i \in A'''$
25:         SET $i, l$ to active
26:     End if
27:     $A' = A' + i$
28: Update($A, A', flow, i$)
29: End if
30: SwitchGrouping($S_{core}, A, U_{upper}$)
31: ValidateAndConsalidate($S_{core}, A, U_{upper}$)
32: Return $A$
The algorithm starts by taking the data center topology; the set of current active switches and ports, the flow to be assigned, and the upper utilization threshold as inputs. After initialization, the network will power on a minimum spanning tree of switches (MST) if the current flow is the first flow to assign, otherwise, the set of all switches that currently powered-on (Set $A'$) will be used. Set $A''$ is the set of all powered-off switches.

For an incoming flow at core switching level, the algorithm randomly selects a switch $i$ from the current set of active switches $A'$. Then, it searches $i$’s routing table to check if there exists an active port $l$ that can lead to the target destination. If so, LinkChecker function will be called to compute the current link load and to verify that the new load will not exceed the capacity of the link (Equation 2.5) and the predefined upper utilization threshold (Equations 2.8 and 2.9). As the flow assigned to a specific link, the current link load will be adjusted accordingly. Meanwhile, any active switch that holds an inactive target link $l$ will be added to set $A''$.

In case no active switch has an active target port that can handle the flow, the algorithm checks if there is an active switch with an inactive target port in set $A''$. If found, it will randomly select a switch from set $A''$ and powered-on the target port $l$ on that switch. If the algorithm failed to find any active switch with a target link that can handle the incoming flow, a new switch $i$ will be randomly selected from set $A''$, powered-on, and a target port $l$ will be activated.

As the incoming flow being assigned, several already assigned flows might be expired. Thus, some switches might be ended up with a light load on its ports and/or replicated switches which might be using different ports. The SwitchGrouping function tries to find any matches for switches grouping and elimination. SwitchGrouping goal is to increase the number of switches to be turned off using two techniques. The first technique involves searching for two replicated candidate switches which are powered on, connected to the same switches, and they use different ports, to group them into one switch by re-allocating all the flows on the lighter switch to the other one and shut it down. For example, in the fat-tree topology with $k = 4$ (Figure 1.b), suppose that the first and second switches at core level (C0 and C1) are powered-on, switch C0 is using ports (0,3) while switch C1 is using port (1). The flows in switch C1 port (1) can be re-allocated to
switch C0 port (1) and switch C1 can be turned-off. The second technique calculates the current traffic load for each switch and starts with the lightest traffic load switch trying to move its traffic load to other switches and turn it off.

For further increase in energy saving, \texttt{ValidateAndConsolidate} function attempts to consolidate links and turns off unused ports in a greedy fashion. It computes active links utilization and chooses candidate links with the lowest utilization for consolidation. The SwitchGrouping and ValidateAndConsolidate functions will assure that the \textit{MST} connectivity property, the link capacity, and the utilization threshold requirements are satisfied. Finally, the final solution of set \textit{A} after the grouping and consolidation processes will be returned.

The heuristic algorithm assures that minimum number of switches and link will be active. Thus, maximizing the energy saving. From network performance point of view, the switches random selection in the heuristic algorithm will distribute the flow load among active links. The link capacity safety threshold maintains extra space within a link to be use in case of sudden traffic surge.

The same algorithm can be used to handle the communications in aggregation level. The only change needed is to use \textit{S}_{Agg} rather than \textit{S}_{C} if it runs over three-tier topology. For aggregation
switching level in fat-tree, the search space for the designated switch will be within a pod not the whole aggregate switches.

To maintain the minimum spanning tree property of reaching all servers all the time, access level switches and ports will not be turned off. It should be noted that the heuristic algorithm is flexible and can modified to handle situations that affect the network traffic such as virtual machine migrations. The worst case computational complexity of the algorithm is $O(S_C \cdot N_s)$, where $S_C$ is the number of switches in the core switching level and $N_s$ is the number of ports per switch.

2.6.1 Performance Bound Analysis

In this subsection we analyze the consolidation performance bound of our proposed heuristic algorithm. Let $OPT$ be the smallest number of switches to be used in a consolidation problem (i.e. the optimal solution).

**Proposition 1.** The number of switched to be used by the heuristic algorithm at each switching level is upper bounded by $\left\lfloor \frac{17}{10} OPT \right\rfloor$.

*Proof.* For a set of flows to be assigned ($F$) to a set of switches at core level ($S_{core}$), the proposed heuristic assigns each incoming flow to the lowest indexed active switch that can handle the flow load based on random proposition. This consolidation similar to applying First Fit ($FF$) method to the bin packing problem. The only difference is that First Fit method searches for a bin (server) sequentially starting from the lowest indexed non-empty bin and our heuristic uses random proposition. Both will pack an item to the first bin that fits in. Since it has been shown that FF has a worst-case result bounded by $\left\lfloor \frac{17}{10} OPT \right\rfloor$ [39], then the performance of the consolidation process of the heuristic algorithm is no worse than $\left\lfloor \frac{17}{10} OPT \right\rfloor$.

It should be noted that if the incoming flows are pre-sorted in a decreasing order, the consolidation process would be similar to the First Fit Decreasing (FFD) method which has a tight upper bound of $\frac{11}{9} OPT + \frac{6}{9}$ [40].
2.7 Load Balancing

In this section, we show how the heuristic algorithm balances traffic loads while turning off a number of network devices. In general, load balancing mechanisms tend to disperse the network traffic among network devices to minimize packet delay, packet loss, and congestion problems [41].

In general, when formulating the energy-saving problem, having a joint objective function to maximize energy saving and load balancing will introduce a contradiction. While energy saving tends to concentrate the load to a small subset of devices, load balancing tries to evenly distribute the load to all links.

In our problem formulation, the main objective is to save data center network energy while setting constraints to preserve network performance. The load balancing requirement can be satisfied (although not perfectly balanced) through the maximum link utilization constraints (Equations 2.8 and 2.9). The maximum link utilization constraints will assure that most of the active links are utilized up to the upper utilization threshold, thus fulfilling the load balancing requirement.

The heuristic algorithm deals with the contradiction by balancing the loads only over active links. It uses a similar idea used by Valiant Load Balancing mechanism (VLB) [17], it randomly selects an active switch and checks if the target link load can handle the demand without violating the upper link utilization constraint. Although the heuristic algorithm – as in VLB – do not guarantee perfect balancing between active links, it satisfies the objective of load balancing by ensuring that any active link load will not be congested.

In fat tree and three-tier topologies, the load balancing mechanism is needed at the core and aggregation levels only. Since the access level should be always active to maintain the server reachability property as a requirement for minimum spanning tree and the fact that each server is connected to one access switch.

Figure 2.2 shows the cumulative distribution function (CDF) for active links load at core and aggregation levels of a fat tree topology for both the proposed scheme (heuristic algorithm) and base case while tasks are distributed equally to all servers. The base case uses round robin load balancing without energy saving mechanism (i.e. all switches and links powered-on all the time).
For the base case, both CDFs illustrate that the active links are not utilized efficiently, most of the active links are only between 25% - 38% for the core level and 15% - 27% for the aggregation level. On the other hand, the heuristic algorithm uses group switching and link consolidation mechanisms to maximize link utilization with respect to the upper utilization threshold and balances the load among active links. In our proposed scheme, most of the links loads are ranging between 55% - 77% of links capacities at core and aggregation levels.

2.8 Performance Evaluation

This section presents the evaluation of our proposed heuristic algorithm. The evaluation is conducted to show that the algorithm achieves a considerable amount of energy saving with minor effect on the network performance. The evaluation divided into two parts: first, we implemented our algorithm using GreenCloud simulator[42], which is based on ns2 [43], to show how the network load will be distributed among core and aggregation switches and how it will affect network energy consumption, network performance metrics, and average imbalance score [44]. The simulation results were compared with the ones obtained by data centers that do not use any energy saving mechanism (Base Case), data centers that use greedy bin-packing energy saving mechanism [16], data centers that use Global First Fit energy saving mechanism [45], and our proposed scheme without load balancing mechanism using synthetic and real packet traces. The base case uses round robin load balancing technique and all network devices are on. Thus, it would produce the best network performance that can be achieved and almost the worst energy consumption. Greedy bin-packing dynamically changes the power state of network devices (links and switches) based on traffic load fluctuation and turns off idle devices. Greedy bin-packing uses full link capacity with no load balancing mechanism. The Global First Fit assigns the current flow in a greedy fashion. The flow will be allocated to the first path that can handle it. To show that the proposed scheme can be applied to various switching-centric topologies, we compared the scheme using both fat tree and traditional tree-tier topologies. Second, we investigate the computation time and power consumption obtained by CPLEX and compare them to the ones achieved by the heuristic
algorithm. This shows the applicability and solution optimality/near optimality of our proposed heuristic algorithm.

2.8.1 Simulation Setup

We have designed two sets of experiments using GreenCloud simulator. The first set is based on fat-tree topology with $k = 30$ port switches. The topology includes 6,750 high computing servers; 225 core switches with 100 Gbps links, 450 aggregation switches with 10 Gbps links, and 450 access switches with 1 Gbps links. The second set is based on three-tier topology, where the topology includes 6,144 high computing servers; 16 core switches with 10 Gbps links, 32 aggregation switches with 1 Gbps links, and 128 access switches with 1 Gbps links. The servers have homogeneous set of resources includes computation, memory, storage resources. Specifically, each server can provide 238,310 MIPS [46], 32 Gigabytes of memory, and 250 Gigabytes of storage. The main topologies parameters considered in the simulations are tabulated in Table 2.2. Also, Table 2.3 shows the power rates of various commodity switches used in the simulation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Three-Tier</th>
<th>Fat Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core switches</td>
<td>16</td>
<td>225</td>
</tr>
<tr>
<td>Aggregation switches</td>
<td>32</td>
<td>450</td>
</tr>
<tr>
<td>Access switches</td>
<td>128</td>
<td>450</td>
</tr>
<tr>
<td>Servers</td>
<td>6144</td>
<td>6750</td>
</tr>
<tr>
<td>Access Links</td>
<td>1 Gbps/3.3μs</td>
<td>1 Gbps/3.3μs</td>
</tr>
<tr>
<td>Aggregation Links</td>
<td>1 Gbps/3.3μs</td>
<td>10 Gbps/3.3μs</td>
</tr>
<tr>
<td>Core Links</td>
<td>10 Gbps/3.3μs</td>
<td>100 Gbps/3.3μs</td>
</tr>
<tr>
<td>Resource - Computational</td>
<td>238,310 MIPS</td>
<td>238,310 MIPS</td>
</tr>
<tr>
<td>Resource - Memory</td>
<td>32 Gigabyte</td>
<td>32 Gigabyte</td>
</tr>
<tr>
<td>Resource - Storage</td>
<td>250 Gigabyte</td>
<td>250 Gigabyte</td>
</tr>
</tbody>
</table>

Clients send high performance computing (HPC) task requests to be executed in the data center. Each task request has a size of 8,500 bytes (6 packets needed). For each request a green scheduler
Table 2.3  Power rates of various commodity switches in Watts

<table>
<thead>
<tr>
<th>Topology</th>
<th>Switch Type</th>
<th>Fixed Power</th>
<th>Port Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three-Tier [47]</td>
<td>Core</td>
<td>2770</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Aggregate</td>
<td>2770</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Access</td>
<td>146</td>
<td>0.42</td>
</tr>
<tr>
<td>Fat Tree [48]</td>
<td>Core</td>
<td>3350</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Aggregate</td>
<td>3184</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Access</td>
<td>1250</td>
<td>13.3</td>
</tr>
</tbody>
</table>

Table 2.4  Network and tasks parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue limit</td>
<td>100 Packets (150 Kbytes)</td>
</tr>
<tr>
<td>Traffic generator</td>
<td>Exponential</td>
</tr>
<tr>
<td>Packet size</td>
<td>1500 byte</td>
</tr>
<tr>
<td>Task. MIPS</td>
<td>100,000</td>
</tr>
<tr>
<td>Task. Memory</td>
<td>1 Gigabyte</td>
</tr>
<tr>
<td>Task. Storage</td>
<td>10 Gigabyte</td>
</tr>
<tr>
<td>Task. Duration</td>
<td>5 Seconds</td>
</tr>
<tr>
<td>Task. Size</td>
<td>8500 byte</td>
</tr>
<tr>
<td>Task. Output</td>
<td>2.5 Megabyte</td>
</tr>
</tbody>
</table>

searches for a target server, which has enough resources to handle the HPC task, from left to right. The green scheduler assigns the requested task to the first server that satisfies its requirements so it aims to consolidate all the tasks to a small subset of servers [47].

Each HPC task consumes 100,000 MIPS, 1 Gigabyte of memory, 10 Gigabytes of storage, and duration of 5 seconds. The output of the HPC task has a size of 2.5 megabytes which is sent from the server back to the client. To simulate traffic surges, each client sending agent has an exponential traffic generator with 1500 bytes packet size. Flows are initiated from clients and their targeted server, and no data traffic generated at any switching level. Each port on a switch has an
independent FIFO queue with a limit of 150 Kilobytes (100 packets). To evaluate the performance of the DCN at different data center loads, the task requests rate is increased accordingly. Table 2.4 summarizes the network and tasks parameters.

2.8.2 Network Energy Consumption

Figures 2.3(a) and 2.3(b) show the network energy consumed by base case (conventional data center), greedy bin-packing, Global First Fit and proposed schemes with/without load balancing with both fat tree and three-tier topologies, respectively. Greedy bin-packing tends to be the most energy saving mechanism. It consolidates routes and uses links at full capacity without setting safety thresholds. Using full link capacity, minimum number of switches will be used, thus, saving a lot of energy. The Global First Fit is able to save energy for low data center loads since it uses the links full capacity and the process of finding a path that can handle flows is easy. On the other hand, as the load increases, data center links became more saturated. This will make finding a path to handle the flows more difficult and time consuming, thus increase power consumption. Greedy bin-packing and Global First Fit are using links at full capacity which in turn will leave the network vulnerable to sudden traffic surges. The proposed scheme tended to save energy while setting up safety thresholds to deal with traffic surges. The results also show that using a load balancing mechanism will cause better energy conservation for the proposed scheme, where the energy consumption is almost similar to the greedy bin-packing even with reserving part of the links capacity as a safety threshold.

The energy consumption of the greedy bin-packing and the proposed schemes with/without load balancing is in direct proportion to the data center load; when the load is low, the number of network devices to turn off is increased so the energy consumption is low. When the load is high, most of the devices need to be active to deal with this load. The base case doesn’t use any energy saving mechanism, thus all network devices are powered-on all the time.

All schemes have the same pattern in both fat tree and three-tier topologies but since fat tree has more network switches and links at core and aggregation levels, it consumes more energy. Since
fat tree has more switches and links with more capacity than three-tier, the number of switches and link to be turned off are much larger. Thus, fat tree can save more energy than three-tier.

2.8.3 Network Throughput

The network throughput evaluates the network transmission capability based on the used approach. Figures 2.4(a) and 2.4(b) present the network throughput of the base case, Global First Fit, greedy bin-packing, and proposed schemes with/without load balancing with fat tree and three-tier topologies at various data center loads.

The base case has the highest practical network throughput since it sends packets over the whole set of links and uses round robin-load balancing mechanism which will minimize transmission time and increase throughput. The proposed scheme with a load balancing mechanism has the closest throughput to the base case, it outperforms greedy bin-packing, Global First Fit, and the proposed scheme without load balancing. This is mainly because the aim of load balancing is satisfied and setting threshold, then less congestion will occur in DCN. For greedy bin-packing and Global First Fit, when the data center load is low, more network devices can be turned off forcing packets to be sent over a small number of links. This will increase the transmission time since more DCN links would be congested, thus, decreasing throughput. Whereas when the data center load is high, all
schemes will transmit packets over almost the same number of links thus the network throughput will be almost the same.

The effect of using the load balancing mechanism in our proposed scheme is significant, it introduces an up to 7% improvement in throughput compared to the same scheme without load balancing. The network throughput with fat tree topology is much higher than three-tier topology; this is because fat tree has more available links with larger capacities at the core and aggregation switches than three-tier. So, three-tier will send packets over less number of links thus decrease throughput.

2.8.4 Average End-to-End Delay

End-to-end delay is the time for a packet to be transmitted across the network from source to destination. The end-to-end delay includes transmission delay, propagation delay, and queuing delay. It is an indication of the overall network performance. Figures 2.5(a) and 2.5(b) show the network average end-to-end delay for all schemes.

Similar to network throughput, the base case tends to have the lowest end to end delay. The proposed scheme with load balancing has the nearest average end-to-end delay to the base case. For the greedy bin-packing, Global First Fit, and the proposed scheme with/without load balancing, when the data center load is low, the packets transmitted over less number of links thus the end
to end delay will increase, as transmission and queuing delays will increase, compared to the base case. When the data center load is high, the proposed scheme will send packets over almost the same number of links as the base case, thus, the end to end delay is almost the same. On the other hand, the greedy bin-packing, Global First Fit, and the proposed scheme without load balancing lack of load balancing mechanism increases the average end-to-end delay significantly. The results show that the Global First Fit has lower end to end delay compared to the greedy bin-packing at low loads, while it became worst as the load increases. This is because the process of finding and assigning paths in Global First Fit is easy and fast when the load is low, but when the load is high, finding paths is more difficult and time consuming which will increase the end to end delay.

The results clearly show the effect of load balancing on end-to-end delay. The end-to-end delay of the proposed scheme with load balancing decreased by up to 14% compared to the same scheme without load balancing. Again, since fat tree contains more links and larger capacities in core and aggregation levels, end to end delay with fat tree is much lower than end to end delay in three-tier.

### 2.8.5 Ratio of Dropped Packets

As data packets transmitted across the data center network, some of them may be lost or dropped and fail to reach their destination due to many reasons. In data center networks with
energy-aware mechanisms, data packets may drop due to link errors, reaching a queue that is already full, or reaching an intermediate link or switch that is turned off. These drops may cause significant network performance degradation as the delay will increase. This is because the data packets drop and their re-transmission happens at the transport layer in the TCP protocol [49]. Figures 2.6(a) and 2.6(b) illustrate the ratio of dropped data packets for all schemes with fat tree and three tier topologies. The results show that the proposed scheme with load balancing provides the nearest drop ratio to the base case. It outperforms the greedy bin-packing, Global First Fit, and the proposed scheme without load balancing for all data center loads with both fat tree and three tier topologies. The proposed scheme maintains the network performance using two techniques; setting up safety threshold to always have extra spaces for incoming packets and adopting load balancing technique for fair load distribution over active links. Using a fat tree, when the load is high, the ratio of drop packets with greedy bin-packing, Global First Fit, and the proposed scheme without load balancing has raise to 2.49%, 2.34%, and 1.95%, respectively, as compared to the proposed schemes’ 1.06%.

The results clearly state that the drop packet ratios with three tier topology are much higher than those with fat tree topology. This is because fat tree has more switches and fatter links capacities compared to three tier, thus, lower data packet drop ratio.
2.8.6 Average Imbalance Score

In section V, we showed how the proposed scheme with load balancing consolidates and balances links to achieve maximum link utilization with respect to the upper utilization threshold. In this part, we evaluated the efficiency of the load balancing mechanism adopted by our proposed scheme using the notion of imbalance score $I$ [44]. Generally, the imbalance score is calculated using the standard deviation (Equation 2.13), where $x_1, ..., x_N$ are the values of a finite data set, $N$ is the cardinality of that set, and $\mu$ is the standard mean.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}, \quad \text{where } \mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$  \hspace{1cm} (2.13)

In data center networks, the average imbalance score for switches ($I_s$) or links ($I_l$) is the standard deviation of the average switch/link utilization across all switches/links in the switching level. Since our proposed scheme minimizes active links and switches, it is essential to compute the average imbalance scores of them for both core and aggregation levels. Suppose that $P_i(t)$ provides the instantaneous link throughput at time $t$, equation 2.14 calculates link utilization over time period when the link is active [50]. $C_i$ is the capacity of link $i$, $T$ is the time interval between measurements, $T_i^*$ is the time when link $i$ is active.

$$u_i = \frac{1}{T_i^*} \int_{t}^{t+T} \frac{P_i(t)}{C_i} \cdot dt$$  \hspace{1cm} (2.14)

Table 2.5 Average imbalance scores

<table>
<thead>
<tr>
<th>Switch Level</th>
<th>Type</th>
<th>Proposed w/o LB</th>
<th>Proposed with LB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Link</td>
<td>0.1723</td>
<td>0.0602</td>
<td></td>
</tr>
<tr>
<td>Core Switch</td>
<td>0.2404</td>
<td>0.0752</td>
<td></td>
</tr>
<tr>
<td>Aggregate Link</td>
<td>0.1833</td>
<td>0.0915</td>
<td></td>
</tr>
<tr>
<td>Aggregate Switch</td>
<td>0.2566</td>
<td>0.0932</td>
<td></td>
</tr>
</tbody>
</table>
Table 2.6 Traces results with Fat tree ($k = 6$)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Case</td>
<td>29399.79</td>
<td>5759.53</td>
<td>1.056</td>
<td>3.11%</td>
</tr>
<tr>
<td>Greedy Bin-Packing</td>
<td>17894.11</td>
<td>3336.56</td>
<td>5.934</td>
<td>6.91%</td>
</tr>
<tr>
<td>Global First Fit</td>
<td>24543.28</td>
<td>3525.19</td>
<td>4.951</td>
<td>5.22%</td>
</tr>
<tr>
<td>Proposed Scheme</td>
<td>18187.45</td>
<td>4934.29</td>
<td>1.472</td>
<td>4.16%</td>
</tr>
<tr>
<td>Proposed Scheme w/o LB</td>
<td>23598.06</td>
<td>4261.63</td>
<td>2.388</td>
<td>4.86%</td>
</tr>
</tbody>
</table>

From equations 2.13 and 2.14, the average imbalance score for links at core switching level can be calculated in equation 2.15, where $|P_C|$ is the cardinality of the set of links at core switching level, $\mu_1$ is the standard mean of all average links load which is calculated in equation 2.16.

$$I_l = \sqrt{\frac{1}{|P_C|} \sum_{i=1}^{|P_C|} (u_i - \mu_1)^2}$$  \hspace{1cm} (2.15)

Where:

$$\mu_1 = \frac{1}{|P_C|} \sum_{i=1}^{|P_C|} u_i$$  \hspace{1cm} (2.16)

Similarly, equation 2.17 shows the imbalance score for switches at core level, where $|S_C|$ is the number of switches at core level, $|N_i|$ is the number of ports within a switch. It calculates each switch utilization based on its links utilization. Equation 2.18 calculates $\mu_2$, which is the standard mean of all average switches utilization at core level.

$$I_s = \sqrt{\frac{1}{|S_C|} \sum_{i=1}^{|S_C|} \left( \frac{1}{|N_i|} \sum_{j=1}^{|N_i|} u_j - \mu_2 \right)^2}$$  \hspace{1cm} (2.17)

Where:

$$\mu_2 = \frac{1}{|S_C|} \sum_{i=1}^{|S_C|} \frac{1}{|N_i|} \sum_{j=1}^{|N_i|} u_j$$  \hspace{1cm} (2.18)

Table 2.5 shows the average imbalance scores for both active links and switches at the core and aggregation levels using a fat tree with $k = 30$ and 30% data center load. The results illustrate that the average imbalance scores for the proposed scheme with load balancing are less than the
Table 2.7 Traces results with Three tier

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Case</td>
<td>3381.67</td>
<td>1499.05</td>
<td>4.839</td>
<td>5.90%</td>
</tr>
<tr>
<td>Greedy Bin-Packing</td>
<td>2058.24</td>
<td>839.715</td>
<td>11.367</td>
<td>11.24%</td>
</tr>
<tr>
<td>Global First Fit</td>
<td>2623.06</td>
<td>866.179</td>
<td>9.024</td>
<td>10.32%</td>
</tr>
<tr>
<td>Proposed Scheme</td>
<td>2191.99</td>
<td>1233.38</td>
<td>5.847</td>
<td>6.97%</td>
</tr>
<tr>
<td>Proposed Scheme w/o LB</td>
<td>2514.33</td>
<td>1074.95</td>
<td>7.317</td>
<td>8.63%</td>
</tr>
</tbody>
</table>

scores for the proposed scheme without load balancing for the links and switches at both core and aggregate levels. The proposed scheme with load balancing uses a VLB like load balancing mechanism which improves the average imbalance scores for links by more than 65% and 50% for core and aggregate switching levels, respectively. Moreover, it also improves the average imbalance scores for switches by more than 68% and 63% for core and aggregate switching levels, respectively. Thus, our proposed scheme balances the load over links and switches efficiently.

2.8.7 Real Packet Traces

The proposed scheme was also evaluated against the base case, Greedy-bin packing, Global First Fit, and the proposed heuristic without load balancing technique using packet traces collected in real data center UNIV1 [51]. The traces include the user application data alongside ARP, ICMP, OSPF, and RIP flows. The flows in the traces have small size ranges (less than 10KB). They were applied to traditional three tier and fat tree topologies ($k = 6$) with 54 servers.

Tables 2.7 and 2.6 show the network energy consumption and network performance metrics (throughput, average delay, and ratio of dropped packets) for all schemes using three tier and fat tree topologies, respectively. The results show that greedy bin-packing achieves the highest energy saving in both topologies since it uses full links capacities. However, greedy bin-packing has the worst network performance as it has the lowest throughput, highest average delay, and highest
packet drop ratio for both topologies. This shows that the greedy bin-packing, although achieves the highest energy saving, is not suitable for data centers with short network flows.

The Global first fit can save around 16.5% and 22.4% of the network energy with fat tree and three tier topologies, respectively, but the network performance is not as good as the proposed scheme nor the proposed scheme without load balancing.

The proposed scheme without load balancing saves around 19.7% and 25.6% of the network energy with fat tree and three tier topologies, respectively. The lack of load balancing mechanism affects the network performance as throughput, average delay, and the ratio of dropped packets are worst compared to the proposed scheme.

The proposed scheme sacrifices part of the network energy that can be saved by setting up links utilization threshold. This threshold alongside the load balancing mechanism preserve the data center network performance. The results clearly show that the proposed scheme has the nearest performance to the base case for all network performance metrics under consideration for both topologies. Furthermore, the proposed scheme is still able to save around 38% and 35% of the network energy with fat tree and three tier topologies, respectively.

2.8.8 More Comparisons

In this subsection we compare our proposed scheme to Virtual machine Placement and Traffic Configuration Algorithm (VPTCA) [52] and Deadline-Constrained Flow Scheduling and Routing (DCFSR) [53] based on the results reported in [52]. VPTCA uses genetic algorithm based VM placement and multiple QoS constrained routing algorithm to save network energy and avoid congestion. In DCFSR, the authors proved that the joint deadline flow scheduling and routing problem is an NP-hard. After that, they proposed an approximation algorithm based on a relaxation and randomized rounding technique. For a fair comparison, we experiment our proposed scheme using the same simulator (i.e. NS2) and same simulation parameters [52]. The algorithms were evaluated using a fat-tree topology ($k = 6$) with 54 servers in terms of DCN energy consumption, average
End-to-End delay, and ratio of dropped data packets with constant bit rates (CBR) 200 and 800 Kbps.

For energy consumption in DCN, the proposed scheme outperforms VPTCA and DCFSR for both light and heavy traffic loads. In particular, the proposed scheme can approximately save around 13% and 28% compared to VPTCA and DCFSR respectively. This is because the proposed scheme uses links consolidation and VLB on active links where as VPTCA relays on optimal initial VM placement of interrelated VMs within a server or a pod to reduce traffic. VPTCA don’t provide any mechanism to deal with congested links nor VM migrations in case of resource usage changes. DCFSR uses full links capacities without using any load balancing mechanism. The flows are prioritized using Early Deadline First (EDF) policy.

For average end-to-end delay and ratio of dropped data packets, satisfying the EDF policy in DCFSR increases its average end-to-end delay and ratio of dropped data packets compared to the proposed scheme and VPTCA. The proposed scheme and VPTCA have similar average end-to-end delay and ratio of dropped data packets. Specifically, the average end-to-end delay using light loads were 0.17, 0.08, and 0.09 milliseconds (ms) and using heavy loads 0.28, 0.27, and 0.25 ms for DCFSR, VPTCA, and the proposed scheme, respectively. Moreover, using light loads, the proposed scheme and VPTCA have almost no dropped data packets while DCFSR has a dropped data packets ratio of around 0.4%. Using heavy loads, the dropped data packets ratios were 1.8%, 0.70%, and 0.78% for DCFSR, VPTCA, and the proposed scheme, respectively.

2.8.9 CPLEX versus Heuristic Algorithm

The MILP formulation is solved using CPLEX. CPLEX results provide the optimal solutions which are taken as a benchmark to evaluate the difference between them and the proposed heuristic algorithm. All experiments were conducted on an identical platform, a Linux machine with 24 Intel Xeon CPUs x5650@2.67 GHz and 47 GB of memory.
To show the validity of optimality for our proposed algorithm, we compared the results with the optimal ones obtained by CPLEX. We found that the final objective values of our proposed algorithm are fairly close to the optimum ones for all the cases under consideration.

Figure 2.7 shows the differences between CPLEX and the heuristic algorithm in terms of power consumption with fat tree topology. The comparison was conducted for data center sizes ranging from 250 hosts \((k = 10)\) to 250000 hosts \((k = 100)\) with data center load of 30\%. Note that \(K\) represents the number ports in a switch at a fat tree and the number of servers of a \(K\)-ary fat tree can be calculated as \(K^3/4\). The results show that the gap between the optimal power consumption and the proposed heuristic algorithm power consumption is less than 4\%. Although the proposed heuristic algorithm can provide solutions slightly less than the optimal, it is much more computationally efficient.

The proposed heuristic algorithm demonstrates high computational efficiency compared to CPLEX as shown in figure 2.8. The growth of computational time for the proposed algorithm increases linearly with the size of the data center, whereas the growth of computational time in CPLEX increases exponentially. There is a slight difference between the solutions obtained by CPLEX and proposed algorithm; however, solving the problem in CPLEX will introduce high computational cost. As the size of the data center goes up, in contrast with the significant boost in computation time for CPLEX, the proposed algorithm solves the problem efficiently. The ratios of
the solving time of CPLEX to that of the heuristic algorithm are considerable, which demonstrates the applicability and scalability for our proposed heuristic especially for large-scale (exascale) data centers.

2.9 Conclusion

The large number of redundant paths and low link utilization in data center networks can be exploited for energy saving. Most research on literature focuses on optimizing energy without any concern about the performance of the network or the ability to handle traffic bursts. In this paper, we conducted a study on saving energy in data center networks while guaranteeing same or similar performance to the original network. We formulate the problem as MILP, where the objective is to minimize energy consumption while introducing load balancing and link utilization thresholds as constraints to maintain network performance and to deal with traffic bursts. The problem solution succeeded to calculate the minimum energy; however, it showed high computational complexity. Thus, for implementation purposes to large data center networks, a suboptimal heuristic algorithm is proposed to solve the problem. The heuristic algorithm switches traffic to a subset of links, turns off unused switches and links, and uses valiant load balancing mechanism on active routes. Simulation experiments for the proposed model under different network topologies and data center
loads show that the proposed model is able to save a considerable amount of energy, improve load balancing for both links and switches with minor effect on network performance.
3.1 Abstract

With the current growth of data centers, improving energy saving is becoming more important to cloud service providers. The data centers architectural design and the advancement of virtualization technologies can be exploited for energy saving. In this paper, we studied the energy saving problem in data centers using virtual machines placement and live migration taking to account the status of the network links load. The problem was formulated as multi-objective integer linear program, which solvable by CPLEX, to minimize the energy consumed by the servers and minimize the time to migrate virtual machines. To overcome CPLEX high computation, a heuristic algorithm is introduced to provide practical and efficient virtual machines placement while minimizing their migration overhead to the network. The heuristic is evaluated in terms of energy consumed and performance using a real data center testbed that is stressed by running Hadoop Hibench benchmarks. The results where compared to the ones obtained by Distributed Resource Scheduler (DRS) and the base case. The results show that the heuristic algorithm can save up to 30% of the server’s energy. For scalability and validity of optimality, the results of the heuristic were compared to the ones provided by CPLEX where the gap difference was less than 7%.

3.2 Introduction

The growth and popularity of cloud computing services is leading toward the rise of large-scale data centers. Current data centers sizes tend to have tens to hundreds of thousands of servers in
order to provide massive and sophisticated services, such as web searching, cloud storage, online social services, and scientific computing [1]. The growth of data centers made it one of the most energy consumed categories in the world. The United States Environmental Protection Agency (EPA) reported that the total electricity used by data centers in 2010 was about 1.3% of all electricity used in the world [2] and it is expected to reach 8% by 2020 [3]. The problem of saving data centers energy is important and challenging for cloud service providers especially with current data center designs and the advancement of virtualization technologies.

Extensive research has been done in the literature to provide solutions to overcome the high data centers energy consumption problem. As the highest source of energy consumption in data centers, most researchers focus their approaches and techniques on finding solutions to the server-side data center energy saving problem [54][55][56][57].

The limitations and drawbacks of the approaches and techniques provided in the literature can be categorized into one of the following five cases: first, they fail to satisfy all server resources requirements (CPU, Memory, Network, and Disk), at the same time, in their solution [58][59][60][61][62]. Such techniques will address only the considered resources leaving others as potential performance bottlenecks. Second, some techniques do not scale to the size of current data centers due to lack of computational efficiency. Third, providing solutions just for the initial placement ignoring the load variations that might happen afterwards [63][64][65][66][67]. Four, the evaluations of some techniques were carried out through simplified simulations running workloads that do not represent real data centers daily workloads. Finally, most techniques try to minimize data center power consumption through virtual machine (VM) migrations while ignoring its effect on network links. This may result in moving VMs over links that are already congested/near congested, leaving the data center network vulnerable to sudden traffic surges. To address this problem, some techniques have proposed a network-aware energy saving techniques [68][69], however, there solutions were mainly focused on the distance between servers (represented by hop count) and the migration cost effect on the source and destination servers without considering the current traffic on network links.
In this paper, we propose a weighted sum multi-objective optimization for data centers energy saving taking into account the effect of virtual machines migration on network links. The multi-objective optimization is part of a framework that monitors the state of the data center by collecting run time utilization data for servers’ resources (CPU, Memory, Network, and Disk). It uses them as an input for the multi-objective optimization. The multi-objective optimization will provide a new virtual machines placement that assure maximum energy saving with minimum effect on the underlying network. Live migration commands will take place to adjust the placement of the virtual machines into their designated destinations based on the optimization solution. Finally, unused servers are set into standby mode.

For large-scale data centers, the running time for the multi-objective optimization, which is solvable via CPLEX, is computationally inefficient. For example, the multi-objective optimization runs for more than 37 hours to provide a solution for a data center with 1500 virtual machines. So, for practical implementation on large scale data centers, a two-phase greedy heuristic algorithm is introduced. The first phase targets finding an initial feasible placement for the virtual machines that satisfies all the resource utilization constraints with minimum migration time. After finding an initial feasible placement, the second phase tries to find an optimal/near optimal solution to efficiently save the data center energy without violating the utilization constraints. To achieve this solution, the heuristic algorithm consolidates virtual machines to a small subset of servers that satisfy their requirements and put unused servers to standby mode. The heuristic searches for the best routes to consolidate virtual machines with minimum effect on the network links. The heuristic continuously monitors the state of each virtual machine and present a new placement if a resource violation occurs or a better solution can be obtained. Live migration moves virtual machines between servers with minimum down time.

To evaluate the efficiency, applicability, scalability, and optimality/near optimality of the proposed framework and the heuristic algorithm, extensive experiments were conducted. The evaluation was divided into two parts: First, experiments on a testbed data center, that is built using VMware vSphere 5.5 suite, were conducted to evaluate performance and energy saving.
Hadoop 2.7.3 multi-node cluster is deployed on the test-bed to mimic real data centers environment, while the testbed is stressed using different workloads from Hibench [70]. The framework is compared to the base case, where no energy saving mechanism is used, as well as VMwares’ Distributed Resource scheduler (DRS). The experiments show that the proposed framework can save energy up to 30% while achieving better performance with minimum effect on the data center network.

Second, to evaluate scalability and validity of optimality of the heuristic algorithm, the solutions of the heuristic algorithm were compared to the optimal ones provided by CPLEX. The comparison shows that the gap between the optimal energy consumption and the ones provided by the heuristic algorithm is at most 7%. Meanwhile, the heuristic algorithm can reduce the computation time significantly compared to CPLEX.

The list of contributions in this paper is as follows:

- We propose a dynamic virtual machine framework with the objective to minimize energy consumption and virtual machines migration effect on the network. The proposed framework actively monitors workload run-time fluctuations and provides dynamic placement solutions.

- A multi-objective optimization formulation for server-side energy saving and time to migrate virtual machines is introduced. The optimization considers all servers resources in its solution (CPU, Memory, Network, and Disk) such that energy wastage and performance bottlenecks caused by resource wastage are eliminated.

- We present a two-stage greedy heuristic algorithm that achieves near-optimal energy saving and low computational complexity. This heuristic is practical solution for large size data centers.

- The heuristic algorithm was evaluated on a real testbed data center. The heuristic was compared with industry leading design VMware’s DRS and the base case to demonstrate the effectiveness of the heuristic algorithm in regard of performance and energy saving.

The rest of the paper is organized as follows. Section 3.3 reviews previous related works. Section 3.4 introduces the proposed system framework. Section 3.5 formulates the multi-objective power
saving problem. Section 3.6 shows the proposed heuristic algorithm. Section 3.7 discusses how the testbed data center was implemented and presents the experimental results, and finally section 3.8 concludes the paper.

3.3 Related works

Many approaches have been proposed to deal with the data centers energy saving problem. A number of researchers proposed designs of new topological structures that provide energy conservation while preserving performance. Examples may include flatted butterfly [20], Pcube [5], Small-World [6], NovaCube [7], 3D Torus based CamCube [21], Nano Data Centers [71], and Proteus [8]. The primary drawback of these new topologies is that they cannot be applied to existing data centers as they require specific hardware and software capabilities.

Other researchers focus on saving energy of the data center network (DCNs). They found optimization problems for current DCNs and propose different techniques and heuristics to solve them. The main idea is to switch the network traffic to a subset of switches and turn off unused devices. Many approaches use such technique such as ElasticTree [16], Carpo [22], REsPoNse [23], GreenTE [14], Merge network [27], and many others [24], [10], [15], [12]. The main concerns in these studies include: the trade-off between energy saving and network performance and how to deal with sudden traffic surges. It should be noted that the amount of energy to be saved by these DCNs techniques is much less compared to the data center server’s energy saving techniques.

Most researchers focus their efforts toward server-side energy saving since the servers are the most energy consuming devices in the data centers and with the advancements of virtualization technologies which provide great opportunities for energy saving. Some studies target only static placement [63][64][65][66][67], these studies consider the initial placement of virtual machines ignoring workloads fluctuation. Other studies suggest live migration for dynamic virtual machine placement [72][73][60]. Such studies ignore the overhead produced by the live migration and its effect on the network. This will lead to placement solutions that require virtual machines to be migrated over congested links or to long distances. For that reason, some researchers propose network-aware
virtual machine placement mechanisms [68][69], they consider the hop-count between the source and destination hosts for migration, the cost of the migration, inter-related virtual machines, and the power consumed during the migration process to minimize the migration overhead and avoiding long distance migrations. The major drawback of these mechanisms is that they don’t consider the current status of the network. Thus, they might migrate virtual machines through routes that are shorter, but already congested or almost congested.

This work overcomes previous studies drawbacks. It takes advantages of the virtualization technologies, uses live migration for dynamic placement while considering all servers resources (CPU, Memory, Network, and Disk). The work also considers the current network status, thus, migrating virtual machines to the most suitable servers with minimum effect on the network. The work is applicable since it was applied to a real data center testbed and evaluated using the widely use Hibench benchmark suite.

3.4 System Model

Figure 3.1 illustrates the three major modules of the proposed framework (Resources measurements module, Multi-objective optimization module, and Next placement module).
The resources measurements module is responsible for the continuous monitoring of the data center and for collecting virtual machines resources utilization (CPU, Memory, Network, and Disk). Furthermore, the module also extracts the network current traffic matrix.

The multi-objective optimization module is a weighted sum integer linear program that takes the collected virtual machines resource utilization and the network traffic matrix as an input. The optimization will provide a Pareto solution that minimizes the energy consumed by the data center, meanwhile minimizes the effect of virtual machines migration on the network. One of the results of this module is a migration matrix, which includes the virtual machines that need to be moved from their current hosting server to another target server.

The last module is the next placement module, which is responsible for sending live migration commands based on the solution provided by the optimization. After all live migration commands completed successfully, the module will put unused servers to standby mode.

3.5 Problem Formulation

In this section, we present the weighted sum multi-objective optimization formulation to minimize the power consumed by servers and the effect of migrating virtual machines on the data center network. The virtual machines migration effect is calculated by finding the time needed for a virtual machine to travel from a source server to a destination server using the current network traffic. Consider a data center $G = (PM, E)$ where $PM$ is the set of all physical machines (servers) and $E$ is the set of all links. The formulation is divided to two parts: minimizing power consumed by servers and the network.

The power consumed by a single physical machine follows the model proposed by [74][75], expressed by equation 3.1. The model shows that the server average power is approximately linear with respect to CPU utilization. The model has been proven to be accurate for large scale data centers.

$$p_{server} = p_{active} + p_{dynamic} \cdot Util^{CPU}$$  (3.1)
Where $P_{\text{server}}$ is the total power consumed by the server, $P_{\text{dynamic}}$ is the dynamic power consumption of the CPU, $Util_{\text{CPU}}$ is the average CPU utilization, and $P_{\text{active}}$ is the power consumption when the CPU is idle.

For the servers’ part, a server $p \in PM$ can provide a set of resources bounded by an upper utilization threshold ($U_p^{\text{CPU}}, U_p^{\text{Mem}}, U_p^{\text{Net}}, \text{and } U_p^{\text{Disk}}$). Let $VM$ be the set of all virtual machines to be hosted by physical machines. Each virtual machine $v \in VM$ requests a specific amount of resources (denoted by $vm_v^{\text{CPU}}, vm_v^{\text{Mem}}, vm_v^{\text{Net}}, \text{and } vm_v^{\text{Disk}}$) to be consumed. A physical machine can host many virtual machines as long as its resource utilization thresholds are not violated.

A physical machine can be turned into standby mode if there is no active $v$ hosted by $p$. A physical machine in standby mode consumes $P_{\text{standby}}$, meanwhile, an active physical machine consumes $P_{\text{active}}$ in addition to the power consumed by each virtual machine hosted by that physical machine $P_{vm}$. We use decision variables $On_i$ to denote the current power status of physical machines (i.e. active or standby mode) and $M_{vp}$ to present the current placement of virtual machines on physical machines.

A virtual machine $v$ can move from one physical machine to another either to minimize power consumption or to solve resource utilization threshold violation. A $g_{vp}$ is a decision variable to denote which virtual machine is migrated and its target physical machine.

With the notations summarized in Table 3.1, we can formulate our problem as a weighted sum multi-objective integer linear program that is solvable by CPLEX as the following: the ILP takes the physical machines $PM$, the virtual machines $VM$, utilization thresholds for resources $U^c_p$, virtual machines current resource utilizations $vm^c_v$, the power of physical machines in active and standby modes $P_{\text{standby}}$ and $P_{\text{active}}$, and the power of each virtual machine $P_{vm}$ as inputs.

The constraints are divided into four categories: placement, power, resource, and network. Equations 3.2 and 3.3 shows the placement constraints, they assure the correct placement of virtual machines on their designated physical machines. Equation 3.2 states that each virtual machine has to be and can only be served by one physical machine. Equation 3.3 illustrates that if a virtual
Table 3.1 Definition of Important Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PM$</td>
<td>Set of all physical machines</td>
</tr>
<tr>
<td>$VM$</td>
<td>Set of all virtual machines</td>
</tr>
<tr>
<td>$E$</td>
<td>Set of all links</td>
</tr>
<tr>
<td>$vm_v^c$</td>
<td>Utilization of resource $c$ by VM $v$</td>
</tr>
<tr>
<td>$U_p^c$</td>
<td>Utilization threshold of resource $c$ at PM $p$</td>
</tr>
<tr>
<td>$i,j$</td>
<td>A link connects two nodes $i$ and $j$</td>
</tr>
<tr>
<td>$P_{standby}, P_{active}$</td>
<td>Power consumed by a PM in standby and active modes, respectively</td>
</tr>
<tr>
<td>$P_{vm}$</td>
<td>Power consumed by a VM</td>
</tr>
<tr>
<td>$bd_e^c$</td>
<td>Bandwidth of link $e$</td>
</tr>
<tr>
<td>$vbd_e$</td>
<td>Bandwidth consumed by virtual machines on link $e$</td>
</tr>
<tr>
<td>$\text{sizeof}(v)$</td>
<td>Size of VM fingerprint</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Tuning variable for weighted-sum</td>
</tr>
<tr>
<td>$M_{vp}$</td>
<td>Virtual machines placement matrix</td>
</tr>
<tr>
<td>$On_p$</td>
<td>Physical machine power mode matrix</td>
</tr>
<tr>
<td>$g_{vp}$</td>
<td>Virtual machines migration matrix</td>
</tr>
</tbody>
</table>

Machine decided to migrate from its current physical machine to a new one, then the value of its current placement $M_{vp}'$ and next placement should change (i.e. $M_{vp}' \neq M_{vp}$).

\[
\sum_{p \in PM} M_{vp} = 1, \quad \forall v \in VM
\]  
(3.2)

\[
M_{vp} - M_{vp}' \times M_{vp} = g_{vp}, \quad \forall v \in VM, p \in PM
\]  
(3.3)

Equations 3.4 and 3.5 present the power constraints, they state that a physical machine operates in active mode if and only if it needs to serve an active virtual machine. Specifically, equation 3.4 assures that a physical machine can be turned into standby mode only if there is no virtual machine
placed on it. Likewise, equation 3.5 shows that a virtual machine can only be placed on an active physical machine.

\[ On_p \leq \sum_{v \in VM} M_{vp}, \quad \forall p \in PM \quad (3.4) \]

\[ M_{vp} \leq On_p, \quad \forall v \in VM, p \in PM \quad (3.5) \]

The power consumed by a physical machine is calculated in equation 3.6. It is based on the status of the physical machine and the virtual machines hosted on that machine.

\[ P(p) = P^{active}On_p + P^{standby}(1 - On_p) + P^{vm} \sum_{v \in VM} M_{vp}vm^CPU_v \quad (3.6) \]

Equations 3.7 - 3.9 introduce the resources utilization constraints. Since virtual machines deployed on a physical machine require some amount of resources, these resources should not exceed a specific utilization level from the resources offered by the physical machines (in this paper, we consider the utilization threshold = 70%). In this formulation, all types of resources are considered (CPU, Memory, Network, and Disk).

\[ \sum_{v \in VM} (M_{vp} \times vm^CPU_v) \leq U_p^{CPU}, \quad \forall p \in PM \quad (3.7) \]

\[ \sum_{v \in VM} (M_{vp} \times vm^Mem_v) \leq U_p^{Mem}, \quad \forall p \in PM \quad (3.8) \]

\[ \sum_{v \in VM} (M_{vp} \times vm^Disk_v) \leq U_p^{Disk}, \quad \forall p \in PM \quad (3.9) \]

For network resources, previous studies focus their effort to the network bandwidth consumed by physical machines, ignoring the bandwidth of all network links, in order to simplify their formulation. In this formulation, we consider the pipe model to express bandwidth constraints on all network links.
Suppose that $\tau$ is the current communication matrix between VMs where $\tau_{v,w}^e$ is 1 if the virtual machines $v$ and $w$ are communicating through link $e$ with bandwidth $\text{bd}_{v,w}^e$. Equation 3.10 calculates bandwidth consumed by all virtual machines communicating through link $e$. Equation 3.11 ensures that the bandwidth passing through link $e$ is less than the capacity of that link by a utilization threshold.

\[
\text{vbd}^e = \sum_{v,w \in \text{VM}} \sum_{p,d \in \text{PM}} \text{M}_{vp}\text{M}_{wd} \min(\text{vm}^N_{v}, \text{vm}^N_{w}) \cdot \tau_{v,w}^e \cdot \text{bd}_{v,w}^e \tag{3.10}
\]

\[
\sum_{e \in E} \frac{\text{vbd}^e}{\text{bd}^e} \leq U^N_e \tag{3.11}
\]

To be able to solve the problem using software solver such as CPLEX, we need to linearize the bandwidth capacity constraint in equation 3.10 which is in bi-linear form [76]. The problem can be linearized by introducing variables $x_{vwpd} \in [0,1]$ that verify the following constraints:

\[
x_{vwpd} \leq \text{M}_{vp}, \quad \forall v, w \in \text{VM}, \quad \forall p, d \in \text{PM} \tag{3.12}
\]

\[
x_{vwpd} \leq \text{M}_{wd}, \quad \forall v, w \in \text{VM}, \quad \forall p, d \in \text{PM} \tag{3.13}
\]

\[
\text{M}_{vp} + \text{M}_{wd} - 1 \leq x_{vwpd}, \quad \forall v, w \in \text{VM}, \quad \forall p, d \in \text{PM} \tag{3.14}
\]

It can be seen that for a given $\text{M}_{vp}, \text{M}_{wd} \in \{0,1\}$, $\text{M}_{vp} \times \text{M}_{wd} = x_{vwpd} \in [0,1]$. So, equation 3.10 can be rewritten as:

\[
\text{vbd}^e = \sum_{v,w \in \text{VM}} \sum_{p,d \in \text{PM}} x_{vwpd} \cdot \min(\text{vm}^N_{v}, \text{vm}^N_{w}) \cdot \tau_{v,w}^e \cdot \text{bd}_{v,w}^e \tag{3.15}
\]

By replacing equation 3.10 with equations 3.12-3.15, the problem becomes ILP. It should be noted that this linearization is valid only if $\text{M}_{vp}$ are integer variables.

The second part of the multi-objective optimization focuses on the time required for a virtual machine to migrate. For a virtual machine $v$ to be migrated, equation 3.16 calculates $\text{Troute}$ which
is the time required for \( v \) to pass through the set of links that form a route between the source host \( p \) and destination host \( d \) (i.e. \( E^* \)). The time for a virtual machine to travel through a link depends on the virtual machine size and current link traffic.

The virtual machine size plays an essential role in calculating the migration time. The virtual machine size represents its state information; this includes its current memory contents and all information that uniquely defines and identifies the virtual machine. The memory contents include the data and instructions of the operating system and the applications that are in the memory. The defining and identification information consist of all the data that maps to the virtual machine hardware elements such as BIOS, I/O devices, CPU, MAC addresses for the Ethernet cards, chip set states, registers...etc. Generally, memory contents are very large compared to the state defining and identification data, thus, we will consider the size of the virtual machine as the size of its memory contents. For a predefined set of routes \( (r_n) \) between the source host \( p \) and destination host \( d \), equation 3.17 chooses the minimum routing time to migrate a virtual machine \( v \) to its target.

\[
T_{route}(v,p,d,E^*) = \sum_{e \in E^*} \frac{sizeof(v)}{U_e^{Net}bd_e - vb_d e}, E^* \subseteq E
\]  

(3.16)

\[
T(v,p) = min(T_{route}(v,p,d,r_1),...,T_{route}(v,p,d,r_n))
\]  

(3.17)

Lastly, equation 3.18 presents the objective function to minimize the power consumed by physical machines as well as the time to migrate virtual machines. The function uses a weighted sum tuning variable \( \gamma \in (0, 1] \). It is an indication of to what extent the cloud provider is willing to sacrifice some the energy saving to decrease migration overhead on the network. \( \gamma \) can be set to large value to indicate that the cloud provider is very keen on energy saving and to a small value when the provider is more concerned about the migration effect on the network. If \( \gamma \) is set 1, the problem
became an energy saving problem without taking the migration overhead into consideration. It is nonsensical to set $\gamma$ to 0, as the problem will not allow any virtual machine migration to occur.

$$\text{Minimize} \quad \gamma \sum_{p \in PM} P(p) + (1 - \gamma) \sum_{p \in PM} \sum_{v \in VM} T(v, p) g_{vp}$$

(3.18)

Since multi-objective integer linear programming is NP-hard, the proposed formulation is not practical for large data center networks. Thus, it can be used as a benchmark tool to evaluate practical heuristic approaches.

### 3.6 Heuristic Approach

To overcome the exponential increase in CPLEX computation time, a heuristic algorithm solving the data center energy-saving problem was developed. In data center environment, workload demands fluctuate frequently. For that reason, heuristic algorithm is preferred to solve our optimization model in real time. Algorithm 2 and 3 illustrate the two-stage heuristic pseudocode, it takes similar inputs as in CPLEX and it is implemented using java programming language with vSphere SDK. The output includes the next placement matrix $M$, the physical machines power mode matrix $On$, and the migration matrix $g$.

Stage 2 starts with the objective of finding an initial placement that satisfies all virtual machines requirements. The algorithm traverses all physical machines searching for a resource utilization violation. A virtual machine placement solution is considered feasible if all virtual machines resource requirements are satisfied and all physical machines consumed resources are within the physical machine resource utilization threshold. If there exists a physical machine that violate utilization threshold for one of its resources, the solution is considered unfeasible and countermeasure operations should be applied to solve the violation.

For a physical machine with a resource utilization violation, the algorithm sorts all the virtual machines currently placed on that physical machine in descending order based on that resource type. Then, finding the set $topVMs$, which includes the virtual machine or the set of virtual machines where their migration will solve the resource violation.
**Algorithm 2** Heuristic Algorithm

1: **Stage 1:** Finding initial feasible solution
2: **Input:** $PM, E, VM, U_{CPU}, U_{Mem}, U_{Net}, U_{Disk}, vm_{CPU}, vm_{Mem}, vm_{Net}, vm_{Disk}, P_{active}, P_{standby}, P_{on}, bd_{e}, vbd_{e}$
3: **Output:** $M, On, g$
4: **for** $PM$ is active **do**
5:  **if** resource exceeds $p \in PM$ utilization threshold **then**
6:  Sort VMs in $p$ in descending of the resource
7:  Find $topVMs$ to solve the violation
8:  **for** $v \in topVMs$ **do**
9:   Find $targetPMs$
10:  **for** $p \in targetPMs$ **do**
11:    Calculate time to migrate $v$ to $p$
12:    Record the lowest time to migrate $v$ to $p$
13:  Migrate VMs with the lowest scores
14:  **if** The violation solved **then**
15:    Return
16:  **if** No feasible solution exists **then**
17:    Adopt alternative strategy to handle violation

**Algorithm 3** Heuristic Algorithm - Continue

1: **Stage 2:** Improving energy-efficiency
2: **loop**
3:  $list$ = Sort $PM$ based on energy consumption
4:  **for** the first $p \in list$ **do**
5:    Check consolidation options
6:  **if** consolidation options are available **then**
7:    Calculate time to migrate $vms$ from $p$
8:    Migrate to the lowest time
9:    Turn $p$ to Standby mode
10:  Update $list$
11:  **else**
12:    Remove from $p$ from $list$
13:  **if** $list$ is empty **then**
14:    return $M, On, g$
For each element in topVMs, a targetPMs set is formed where each element in it represents a physical machine that is a potential destination. To avoid high computational running time, since this set might contain large number of elements considering today’s data centers sizes, the search for a target physical machine will be within a rack or a pod. Next, the migration time for each potential destination is measured based on the network current traffic and the virtual machine size, and the lowest migration time is recorded such that there is no conflict on the network routes. Finally, the virtual machine(s) with the lowest recorded time will be migrated to its destination. The process will be repeated for all physical machines until a feasible solution is reached. If no feasible solution exists, other alternative strategies can be adopted to solve the resource utilization violation.

After an initial feasible solution is found, stage 3 of the heuristic algorithm aims to improve the current energy consumption while presenting minimal overhead to the network links. The adopted strategy moves virtual machines to a subset of physical machines and puts unused physical machines to standby mode.

This stage begins by calculating the power currently consumed by each physical machine and sorting them, into list, in ascending order based on their power consumption. For the first physical machine p in the ordered list, the algorithm searches for available consolidation options in order to move all virtual machines currently residing on that physical machine. The consolidation options are the set of physical machines that can handle the virtual machines currently hosted by p without violating the resource utilization thresholds (equations 3.7-3.9). For large data centers, this set might become very large, so the consolidation options candidates are limited to the hosts within p’s pod or rack. If found, for each virtual machine, calculate the migration time to move it to the target physical machine with the lowest migration time. Note that the communication matrix will be updated after each vm migration to avoid conflicts on the network route. When all virtual machines hosted by the physical machine migrated to their target physical machine(s), the physical machine will be set to standby mode, removed from list, and list will be updated. If no consolidation option found, the physical machine p will be removed from list. This iterative process is repeated
until list is empty; thus, no further improvement can be made to the solution and the algorithm returns the next placement matrix, the physical machines power matrix, and the migration matrix.

Overall, the computation complexity for the proposed heuristic algorithm is $O(PM^2VM \log PM \log VM)$.

### 3.7 Evaluation and Experimental Evaluation

This section presents the evaluation of our proposed heuristic algorithm. The evaluation is conducted to show that the proposed heuristic algorithm is efficient, applicable, scalable, and can achieve considerable amount of energy saving to the data center while maintaining network performance. The evaluation is divided into two parts: in the first part, a three-node data center testbed is built and stressed through fluctuation workloads using Hibench 6.0. Hibench is a benchmark developed by Intel to evaluate the performance of MapReduce jobs running in data centers for both Hadoop and Spark. As benchmark loads are running, the proposed heuristic will adjust the locations of the virtual machines in order to avoid physical machines resource utilization limit violation and to save energy.

The heuristic algorithm results were compared to the ones obtained by the VMware’s Distributed Resource Scheduler (DRS) [57]. DRS is used to manage the placement of the virtual machines within a cluster. DRS focuses on balancing the load across all physical machines by calculating the cluster imbalance score $I_c$ and made VM migration decisions to minimize or maintain it under a given threshold. The imbalance score is the standard deviation of the load over all physical machines. DRS periodically (every 5 minutes by default) invokes a greedy hill-climbing algorithm to calculate the cluster imbalance score and make migration decisions.

An extended feature of DRS called Distributed Power Management (DPM)[44] is used to save energy by moving virtual machines from lightly loaded physical machines and puts physical machines with no virtual machines into standby mode. DPM periodically search each physical machines' resources utilization and provide recommendations for energy saving if the load for a physical machine is lower than a predefined threshold. Furthermore, the heuristic is also compared to the base
case, where no virtual machine migration is allowed. The evaluation was conducted using the same workload for all designs and each test was repeated three times and the average was recorded.

In the second part, we investigate the computation time and energy consumption of the optimal solutions obtained by CPLEX 12.7 and compare them to the ones achieved by our proposed heuristic. This will prove the scalability and optimality/near optimality of our proposed heuristic algorithm.

3.7.1 Testbed Setup

The testbed data center is built using VMware vSphere suite 5.5 to prove the applicability, energy efficiency, and performance effectiveness of the proposed framework.

Currently, the testbed is configured using three physical machines to host virtual machines. Each physical machine is equipped with a quad-core 3.4 GHz Intel i7 processor and 32 GB of memory. The physical machines use a 1 Gbps private network for communication and virtual machines migrations. Figure 3.2 shows the testbeds’ network topology. An ESXi 5.5 hypervisor is running on each physical machine for deploying and serving virtual machines. Note that the energy consumed by the physical machines were measured using Kill a watt [77] energy monitoring device.

MainCenter is a virtual server that has a vCenter tool to manage and control all events happened on the data center such as initiating migration commands and enter/exit a physical machine from
standby mode. Furthermore, the vCenter collects runtime performance measurements and saves them to a Microsoft SQL Server 2005 database. Another virtual server called DNS provides domain name, Active directory domain, and network storage services. Both virtual servers are running Windows server 2008 R2 with 4 GB of memory and 60 GB of storage.

Furthermore, a 13 Linux Ubuntu 16.04 LTE virtual machines are deployed in the data center testbed. They are equipped with an iSCSI network storage that is accessible by all physical machines. The virtual machines share a 1 TB iSCSI storage and they use 1 Gbps vMotion network for live migration. A Hadoop multi-node cluster is configured using Apache 2.7.3 to evaluate the testbed using MapReduce benchmarks. The Hadoop cluster runs in default settings and it includes one master node (name node) and 12 slave nodes (data nodes). Each node has a 2 GHz CPU capacity, 4 GB of memory, and 40 GB of storage.

To increase the utilization of a single node, Docker containers is used. Dockers containers help in running distributed applications within a Linux instance. This technology is becoming popular for applications in cloud environment since there is no need for deploying and managing new virtual machines, thus, reducing overhead. The estimation of a virtual machine resource usage is based on its history resource usage. Finally, it should be noted that we use a one-minute threshold for initiating virtual machines migrations.

### 3.7.2 Testbed Results

Figure 3.3 shows the network bandwidth consumed by the testbed servers when managed by VMware’s DRS. For a 15 minutes period, the results illustrate that servers managed by DRS show large variation in network bandwidth consumption for each server. For example, server 3 only consumes up to 13% of its network bandwidth. Meanwhile, server 2 consumes no lower than 51% and up to 82% of its network bandwidth. This large variation happens since DRS makes migration decisions to maintain the imbalance score $I_c$, which is mainly based on CPU and memory resources, while network resources are being ignored.
This situation might produce a bottleneck that affects the performance of the data center, especially if multiple virtual machines that require high network bandwidth and low CPU and memory resources are placed on the same server. Thus, the server will have available CPU and memory resources, but it will not be able to host new virtual machines. This problem is known as resource contention problem.

On the other hand, Figure 3.4 shows the network bandwidth consumed by the testbed servers when managed by our heuristic algorithm. Starting with high variations of network bandwidth consumption, the heuristic algorithm detects this variation and provides new placement to overcome
any potential bottleneck. The heuristic algorithm invokes migration commands to balance the network resource (it also considers balancing CPU, memory and disk resources) with minimum overhead since the virtual machines are migrated to the target machine with minimum migration time. In Figure 3.4, within 5 minutes, server 1 network bandwidth consumption is decreased from 73% to around 23% and the heuristic algorithm maintains the network bandwidth consumption balancing between the testbed servers afterwards.

For the evaluation of the Hadoop cluster configured on the testbed, Hibench benchmarks: Wordcount, TeraSort, PageRank, and Kmeans are used [70]. Figure 3.5 shows the energy consumed by the testbed when running each benchmark for the base case, DRS, and the heuristic algorithm. The results clearly show that the heuristic algorithm outperforms the base case and DRS for all benchmarks. For the Wordcount benchmark, which is MapReduce job used to count the occurrence of each word in a randomly generated text, the heuristic algorithm and DRS have almost the same energy consumption while the base case consumes more energy. The energy saved by the heuristic is small, this is because of the overhead introduced by the virtual machines live migration. Tera-sort benchmark sorts a randomly generated text. The heuristic algorithm consumes around 57.6 KJ while DRS and base case consume 64.8 and 83 KJ, respectively. To evaluate web searching, PageRank benchmark is used. It is an implementation of Google’s web page ranking algorithm.
Figure 3.6  Testbed computational running time.

The PageRank MapReduce job is to rank 500000 web pages using three iterations. Using PageRank benchmark, the heuristic algorithm consumes less energy than DRS and the base case.

For all benchmarks, it is obvious that the base case cannot deal with the resource contention problem, thus, it needs more time to finish the jobs and consumes more energy. Furthermore, DRS reacts periodically to detect and solve the resource contention problem (every 5 minutes), so the servers need to wait until the DRS is invoked. These servers will suffer extra energy to be consumed. The heuristic algorithm detects the contention problem and solve it quickly via live migration. The live migration will introduce extra overhead with small performance degradation for the application on the migrated virtual machine during the migration process. The heuristic algorithm migrates virtual machines with minimum migration time to reduce such overhead. Finally, Kmean benchmark is a MapReduce job for machine learning. The job is to cluster 20 dimensions, 20 million samples into 5 clusters with $K = 10$ and maximum iterations is 5. The heuristic algorithm consumes around 182 KJ of energy compared to 206.4 KJ for DRS and 236.9 KJ for the base case.

Similarly, Figure 3.6 shows the computational running time for the base case, DRS, and the heuristic algorithm when running Hibench benchmarks: WordCount, TeraSort, PageRank, and Kmeans. Like the consumed energy, the heuristic algorithm is more efficient than the base case and DRS in terms of computational efficiency for all tested benchmarks.
Table 3.2  Comparison of the number virtual machines migrations

<table>
<thead>
<tr>
<th>Approach</th>
<th>Base Case</th>
<th>DRS</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of VM migrations</td>
<td>0</td>
<td>23</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 3.7  Comparison of energy consumption between the proposed heuristic algorithm and CPLEX.

The number of virtual machines migrations is an indication of the data center stability. In this experiment, we ran the Hibench workloads (Wordcount, TeraSort, PageRank, and Kmean) all at once, and record the number of virtual machine migrations. Table 3.2 show the number of virtual machine migrations for the testbed using the Base case, DRS, and the Heuristic algorithm. Since the Base case do not use VM migration, the number of migrations is 0. The heuristic algorithm considers all resources in its solution; thus, it needs a smaller number of migrations compared to DRS.

From the evaluation, it could be concluded that the heuristic algorithm is efficient in terms of energy saving and performance. Also, the heuristic algorithm can detect and solve contention problems for all server resources. Lastly, it should be noted that the testbed is using shared storage and storage migration is not implemented in this framework yet.
3.7.3 CPLEX versus Heuristic Algorithm Results

We implement the heuristic algorithm using Java programming language. Furthermore, a linearized version of the proposed multi-objective formulation is solved using CPLEX 12.7 [78]. CPLEX results provide the optimal solutions that are taken as a benchmark to evaluate how optimal are the solutions provided by the proposed heuristic algorithm. The experiments were conducted using synthetic data on an identical platform; a Linux machine with 32 Intel Xeon CPUs E5-2650 @ 2.00 GHz and 256 GB of memory.

To show the validity of optimality for our proposed heuristic algorithm, we compared the difference gap between the results provided by the heuristic algorithm with the optimal ones obtained by CPLEX. We found that the energy consumption values of our proposed algorithm are fairly close to the optimum ones for all the cases under consideration.

Figure 3.7 shows the differences between CPLEX and the heuristic algorithm in terms of energy consumption. The comparison was conducted for data centers hosting virtual machines ranging from 10 VMs to 1500 VMs (3 to 300 PM see Table 3.3 for details) with 5 minutes running period. The results show that the gap between the optimal energy consumption and the ones obtained by the heuristic algorithm is less than 7% for all cases. For example, a data center hosting 750 virtual machines will consume around 6461.8 KJ in the optimal case, while using the heuristic algorithm it will consume around 6925 KJ with a 6.7% difference gap between them. Although the proposed heuristic algorithm provides solutions that are slightly less than the optimal, it is much more computationally efficient.

The proposed heuristic algorithm demonstrates high computational efficiency compared to CPLEX as shown in Table 3.3. The growth of computational time for the proposed algorithm increases linearly with the size of the data center, whereas the growth of computational time in CPLEX increases exponentially.

There is a slight difference between the solutions obtained by CPLEX and proposed algorithm; however, solving the problem in CPLEX will introduce high computational cost. As the size of the data center goes up and hosts more VMs, in contrast with the significant boost in computation
Table 3.3  Comparison of computational time between the proposed heuristic algorithm and CPLEX (in seconds)

<table>
<thead>
<tr>
<th>No. of VMs</th>
<th>No. of PMs</th>
<th>Heuristic</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>0.004</td>
<td>0.09</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0.084</td>
<td>1.02</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0.17</td>
<td>4.25</td>
</tr>
<tr>
<td>250</td>
<td>60</td>
<td>0.457</td>
<td>87.92</td>
</tr>
<tr>
<td>500</td>
<td>120</td>
<td>2.06</td>
<td>946.48</td>
</tr>
<tr>
<td>750</td>
<td>180</td>
<td>4.683</td>
<td>4873.08</td>
</tr>
<tr>
<td>1000</td>
<td>240</td>
<td>14.158</td>
<td>21816.22</td>
</tr>
<tr>
<td>1500</td>
<td>300</td>
<td>54.454</td>
<td>136040.35</td>
</tr>
</tbody>
</table>

time for CPLEX, the proposed algorithm solves the problem efficiently. For example, CPLEX needs more than 940 seconds to find the optimal solution for a data center hosts 500 VMs. Meanwhile, the heuristic algorithm needs only 2 seconds. Moreover, for a data center that hosts 1500 VMs, CPLEX needs more than 37 hours to provide the optimal solution while the proposed heuristic algorithm can obtain a solution in less than one minute.

The ratios of the solving time of CPLEX to that of the heuristic algorithm are considerable. They demonstrate the applicability and scalability for our proposed heuristic especially for large-scale (exascale) data centers.

3.8 Conclusion and Future Works

The current growth of data centers sizes make energy saving problem important for cloud service providers. The development of virtualization technologies provides opportunities for energy saving. In this paper, we present a framework for managing and controlling virtual machines placement on physical servers to reduce the energy consumed by data centers. Furthermore, the framework considers the current status of the network when making migration decisions. The problem was formulated as a multi-objective ILP to reduce consumed energy and minimizing migration time. The problem solution succeeded to calculate the minimum energy and migration time; however, it
showed high computational complexity. Thus, for implementation purposes to large data centers a two-stage heuristic algorithm is proposed. The heuristic monitors the physical machines and virtual machines resources and reacts if a resource threshold violation occurs or a better solution is found considering the network status. The heuristic algorithm was evaluated using a real data center testbed against DRS and the base case in terms of performance and energy saving. For the cases under consideration, it was found that the heuristic algorithm can save energy while maintaining performance and introducing minimum virtual machines migration overhead to the network links. Moreover, the heuristic algorithm solutions were compared to the optimal ones obtained by CPLEX. The solutions were fairly close to the optimum ones and the heuristic algorithm provide a much better computational running time.

For future works, the proposed framework can be evaluated by a larger testbed with different virtualization platforms such as Xen. Furthermore, the proposed formulation can be part of a joint optimization that saves network and server sides of the data center.
CHAPTER 4. PREDICTION-BASED JOINT ENERGY SAVING OPTIMIZATION FOR VIRTUALIZED DATA CENTERS

A paper to be submitted to IEEE Transactions on Network and Service Management
Motassem Al-Tarazi and J. Morris Chang

4.1 Abstract

Today’s data center tend to have tens to hundreds of thousands of servers to provide massive and sophisticated services. Statistically, data center and data center networks DCNs remain highly under-utilized which can be exploited for energy-saving. In this paper, we have studied energy-saving problem for the network and server sides of the data center. The problem was formulated as MILP that is solvable by an optimization software to jointly minimize the energy consumed by the servers and DCN. To overcome the optimization software high computational time, a heuristic algorithm to provide practical and efficient solution for the joint MILP is introduced. The heuristic algorithm has two stages where first it uses the virtual machines (VM) and servers predicted resource utilization to provide VM consolidation algorithm. The second stage uses an abstract performance aware network flow consolidation. Simulation experiments using CloudsimSDN were conducted to validate the heuristic using real traces from Wikipedia in terms of energy consumption and average response time. The results show that the heuristic can save servers and network devices while maintaining performance.

4.2 Introduction

In the era of cloud computing, data centers are growing in size leading toward the rise of large-scale data centers. One of the major concerns in cloud computing is the huge electricity consumption in the cloud data centers. According to the United States Environmental and Protection Agency
the total electricity used by data centers in 2010 was about 1.3% of the all consumed electricity in the world and expected to reach 8% by 2020. Another report estimates the annual total energy costs of data centers in US alone to reach $13.7 billion by 2020 [79]. The major consumers in almost any data center includes servers, cooling systems, and data centers networks (DCNS) [80][81]. The energy consumption percentage for each major consumer can be estimated as follows: servers (40-60%), cooling systems (15-30%), and DCNs (5-15%). This percentage breakdown can change from a data center to another.

The architectural design of data centers is usually built to handle worst-case workload scenarios, which results in low average utilization for servers and rarely reaches its peak power. For example, Fan et al. [82] reported that, over the course of six months, a group of 5,000 servers under study at Google never exceeded 72% of their aggregate peak power. Due to this low utilization, server consolidation techniques have been proposed to increase server utilization and reduce energy consumption by putting unused servers to standby mode.

Similarly, the design of DCNs accommodates peak loads in most reliable way without taking energy saving into consideration. Data center networks are built with many redundant links and heavily over-provisioned link bandwidth to handle link failures and traffic bursts. Although current data centers design increases reliability, it also decreases energy efficiency since all network devices are powered-on all the time with minimal link utilization. Statistics showed that most of the network devices are under-utilized, where the typical utilization of a DCN is only 30% [4]. DCNs’ over-provisioning and under-utilization can be exploited for energy saving research. Routes consolidation techniques are proposed to turn the network load to a minimal subset of network devices. Then it puts unused devices to sleep mode or shut them down to minimize the overall network power consumption. Most research efforts focus on power consumption of server within data center and power consumption of data center networks separately. Considering power consumption of servers using server consolidation with taking DCN into account ignores the effect of virtual machine migration on DCN and increases the chances of traffic congestion which leads to network performance degradation. On the other hand, considering DCN power consumption alone
using routes consolidation ignores that the traffic might be affected by other events such as virtual machine migration.

In this paper, we studied the problem of saving servers and network energy consumption in virtualized data centers while maintaining their performance. We formulate the problem as a joint mix integer linear program to minimize the total servers and network energy as main objective. Moreover, the problem was constrained by network performance requirements, such as maximum link utilization and safety margin threshold for network links and servers resources.

The joint optimization is part of a framework that monitors the state of the data center by collecting and predicting run time utilization data for servers resources (CPU, memory, network, and disk) and network traffic. It uses them as an input for the joint optimization. The joint objective optimization will provide a new virtual machines placement and flow routing matrix that assure maximum data center energy saving while maintaining performance. Live migration commands will take place to adjust the placement of the virtual machines into their designated destinations based on the optimization solution. Finally, unused servers are moved to standby mode and switches are turned off. For large-scale data centers, the running time for the joint optimization, which is solvable via optimization software, is computationally inefficient. Thus, a heuristic algorithm for saving data center and network energy is proposed. The proposed heuristic has two stages; first at the servers side, the virtual machines initial placed using First Fit Decreasing. After that, the heuristic uses the resource predictions to solve resource utilization violations and save more energy. The second stage includes the use of an abstract performance aware flow routing and consolidation.

The proposed heuristic was evaluated using CloudsimSDN simulator using traces collected from Wikipedia page view statistic. The results show that the proposed algorithm can save significant amount of energy in both servers and network sides while maintaining performance represented by average response time.

The rest of the paper is organized as follows. Section 4.3 reviews previous related works. Section 4.4 introduces the proposed system framework. Section 4.5 shows the prediction model used. Section 4.6 formulates the joint power saving problem. Section 4.7 shows the proposed
heuristic algorithm. Section 4.8 presents the simulation results and Section 4.9 concludes the paper.

4.3 Related works

Many approaches have been proposed to deal with the data center server-side energy saving using servers virtualization and virtual machines consolidation as servers are the most energy consuming devices, thus, providing great opportunity for energy saving [68][69][83][84][60]. For example, [84] propose a multi-objective optimization that consolidate the virtual machines into subset of servers taking into account the effect of virtual machine migration on network links. They propose a two stage heuristic algorithm where the first stage finds an initial feasible placement and the second stage triggered periodically to try to consolidate virtual machines into smaller set of servers to save more energy.

Other researchers focus on saving energy of the data center network (DCNs). They found optimization problems for current DCNs and propose different techniques and heuristics to solve them. The main idea is route consolidation, which try to switch the network traffic to a subset of switches and turn off unused devices. Many approaches use such technique such as ElasticTree [16], Carpo [22], REsPoNse [23], GreenTE [14], Merge network [27], and many others [24; 10; 15; 12].

ElasticTree [16] proposed a power manager that adjusts the active switches and links to satisfy dynamic traffic loads. Carpo [22] introduced a correlation-aware power optimization algorithm, it dynamically consolidates traffic loads into a minimal set of switches and links and shut down unused devices. REsPoNse [23] discussed the trade-off between optimal energy saving and scalability. It identifies a few critical routes offline, installs them to routing tables, then runs an online simple scalable traffic engineering to activate and deactivate network devices.

Recently, many researchers start to consider energy saving joint optimization for servers and DCNs. [85] studied the network routing and VM placement problem jointly to minimize traffic cost in DCN. They propose an online algorithm based on Markov approximation to find a near optimal solution within a feasible time.
VMPlanner [86] optimizes VM placement and network routing. They group VMs with high mutual traffic and assign them to the same rack. After that, traffic flows within a rack are consolidated to turn off unused switches.

PowerNets [87] considers finding optimal VM placement considering both servers and network resources and the correlation between VMs. They calculates the correlation coefficient between traffic flows and applied them for VM and traffic consolidation.

The proposed algorithm uses resource predictions for CPU, Mem, Dist, and network resources. Jointly optimize both servers and network sides of the data center.

4.4 System Model

Figure 4.1 illustrates the major modules of the proposed framework (Resources measurements and prediction module, Joint Optimization module, and the Next placement and Power module). The resources measurements and prediction module is responsible for the continuous monitoring of the data center and for collecting virtual machines and hosts resources utilization and predictions (CPU, memory, network, and disk). Note that local agents at each host calculates the resource utilization predictions for the host and each virtual machine. Furthermore, the Resources measurements and prediction module also extracts the network current traffic matrix.

The joint optimization module is a mix integer linear program that takes the collected virtual machines and hosts resource utilization and predictions and the network traffic matrix as an input. The optimization will provide a solution that minimizes the energy consumed by the data center while maintaining performance. The results of this module are the new placement matrix, the flows routing matrix, and the devices power matrix.

The last module is the next placement and power module, which is responsible for sending live migration commands based on the solution provided by the optimization. After all live migration commands completed successfully, the module will turn unused servers to standby mode and turn the switches off.
4.5 Resource Measurement and Prediction

The first module in our proposed framework is the resource measurement and prediction module. It periodically collects hosts and virtual machines resources utilization as well as virtual machines predictions from local agents. Using these information, the module calculates hosts resource utilization predictions using linear regression for all resource types (CPU, Mem, Disk, and Net). Likewise, the local agent at each physical host collects virtual machine resource utilization and calculate prediction using linear regression.

Linear regression is a popular statistical approach to estimate the relationship between one or more input and one output. Linear regression approximate the regression function which represents a straight line. The regression function for the linear regression can be expressed as:

\[ y = \beta_0 + \beta_1 x \]  

(4.1)

Where \( \beta_0 \) and \( \beta_1 \) are the regression coefficient. They indicate the goodness of the fit and how well it predicts the output of \( y \). The popular least square method is used to minimize the residuals.

The resource measurement and prediction module will categorize the hosts and virtual machines into one of the following: Overloaded, predicted to be overloaded, normal, predicted to be under-
loaded, and underloaded. These information and the resource utilization will be used as an input for the optimization and the heuristic algorithms to make better decisions.

4.6 Problem Formulation

Consider a data center $G = (P \cup S, E)$ where $P$ is the set of hosts, $S$ is the set of switches and $E$ is the set of links that connect switches and hosts and switches together. Each link $(i, j) \in E$ has a maximum capacity denoted by $C_{i,j}$, where $C$ is the bandwidth capacity matrix.

The power consumed by a single physical machine follows the model proposed by [74][75], expressed by equation 4.1. The model shows that the server average power is approximately linear with respect to CPU utilization. The model has been proven to be accurate for large scale data centers.

$$
\epsilon_{\text{server}} = \epsilon_{\text{Pactive}} + \epsilon_{\text{Pdynamic}} \cdot \text{Util}_{\text{CPU}} 
$$

(4.2)

Where $\epsilon_{\text{server}}$ is the total power consumed by the server, $\epsilon_{\text{Pdynamic}}$ is the dynamic power consumption of the CPU, $\text{Util}_{\text{CPU}}$ is the average CPU utilization, and $\epsilon_{\text{Pactive}}$ is the power consumption when the CPU is idle. A server $p \in PM$ can provide a set of resources bounded by an upper utilization threshold ($U_{p}^{\text{CPU}}$, $U_{p}^{\text{Mem}}$, $U_{p}^{\text{Net}}$, and $U_{p}^{\text{Disk}}$). Let $VM$ be the set of all virtual machines, and $v \in VM$ must be hosted by a physical machine $p$, denoted by $H(p, v) = 1$. Each virtual machine $v$ requests a specific amount of resources (denoted by $vm_{v}^{\text{CPU}}$, $vm_{v}^{\text{Mem}}$, $vm_{v}^{\text{Net}}$, and $vm_{v}^{\text{Disk}}$) to be consumed. A physical machine can host many virtual machines as long as its resource utilization thresholds are not violated. Note that in this paper we use host, physical machine, and server interchangeably.

A port link can be turned-off if there is no traffic on the link, and a switch can be turned-off if all its ports are turned-off. The power consumed by a single switch $s \in S$ consists of fixed power $\epsilon_{\text{Sactive}}$, which consumed by components like (chassis, fans, etc), and ports power $\epsilon_{\text{Port}}$. The power saving gained from turning off a single port is $\epsilon_{\text{Port}}$, and from turning off an entire switch is $\epsilon_{\text{Sactive}} + \sum_{l \in N_{s}} \epsilon_{\text{Port}}$. 
We use $On(i)$ and $On(i,j)$ as decision variables to denote that switch (or a server) $i$ and link $(i,j)$ are active or not. Assuming $F$ is the set of all flows in DCN. A flow $f \in F$ consists of a source node $src(f)$, a destination node $dest(f)$, and the required bandwidth $bw(f)$. $route(f,(i,j)) = 1$ denotes that a flow $f$ is using link $(i,j)$, where $(i,j) \in E$.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>Set of all switches</td>
</tr>
<tr>
<td>$P$</td>
<td>Sets of all hosts</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of all ports</td>
</tr>
<tr>
<td>$E$</td>
<td>Set of all links</td>
</tr>
<tr>
<td>$D$</td>
<td>Set of all traffic Demands</td>
</tr>
<tr>
<td>$C$</td>
<td>Capacity Matrix for all links</td>
</tr>
<tr>
<td>$VM$</td>
<td>Set of all virtual machines</td>
</tr>
<tr>
<td>$N_i$</td>
<td>Ports in switch $i$</td>
</tr>
<tr>
<td>$i,j$</td>
<td>A link connects two nodes $i$ and $j$</td>
</tr>
<tr>
<td>$\epsilon_{sactive,\epsilon_{port}}$</td>
<td>Energy consumed by an active switch and port</td>
</tr>
<tr>
<td>$\epsilon_{Pactive,\epsilon_{Pstandby}}$</td>
<td>Energy consumed by a host in active and standby modes</td>
</tr>
<tr>
<td>$\epsilon_{vm}$</td>
<td>Energy consumed by a virtual machine</td>
</tr>
<tr>
<td>$vm_c$</td>
<td>Utilization of resource $c$ by VM $v$</td>
</tr>
<tr>
<td>$U_c^p$</td>
<td>Utilization threshold of resource $c$ at host $p$</td>
</tr>
<tr>
<td>$On(i,j)$</td>
<td>A Link $(i,j)$ is on or off</td>
</tr>
<tr>
<td>$On(j)$</td>
<td>A Switch or host $(j)$ is on or off</td>
</tr>
<tr>
<td>$H(p,v)$</td>
<td>A VM $v$ hosted by a host $p$</td>
</tr>
<tr>
<td>$route(f,(i,j))$</td>
<td>Flow $f$ is routed using link $(i,j)$</td>
</tr>
<tr>
<td>$u_{i,j}$</td>
<td>Utilization of link $i,j$</td>
</tr>
</tbody>
</table>

With the notations summarized in Table I, we can formulate our problem as a mixed integer linear program that is solvable by an optimization software as the following: the MILP takes the
data center $G = (P \cup S, E)$, utilization thresholds for links and host resources $U^c_p$, virtual machines current resource utilizations $vm^c_v$, virtual machines predicted resource utilizations $vm^c_{v_t}$, the set of all flows $F$, the capacity matrix $C$, and the power specifications for the physical machines, virtual machines, and switches.

Equation 4.2 is the objective function. It minimizes the data center power consumption function $P(x)$ for every host and switch.

$$\text{Minimize } \sum_{p \in P} H_{\text{pow}}(p) + \sum_{s \in S} S_{\text{pow}}(s)$$  \hspace{1cm} (4.3)

The constraints are divided into three categories: servers constraints, network constraints, and links constraints. Equations 4.3-4.4 present servers power constraints, they state that a physical machine operates in active mode if and only if it needs to serve an active virtual machine. Specifically, equation 4.3 assures that a physical machine can be turned into standby mode only if there is no virtual machine placed on it. Likewise, equation 4.4 shows that a virtual machine can only be placed on an active physical machine.

$$On(p) \leq \sum_{v \in VM} H(p,v), \quad \forall p \in P$$  \hspace{1cm} (4.4)

$$H(p,v) \leq On(p), \quad \forall v \in VM, p \in P$$  \hspace{1cm} (4.5)

The power consumed by a physical machine is calculated in equation 4.5. It is based on the status of the physical machine and the virtual machines hosted on that machine.

$$H_{\text{pow}}(p) = \epsilon_{\text{active}} On(p) + \epsilon_{\text{standby}} (1 - On(p)) + \epsilon_{vm} \sum_{v \in VM} H(p,v)vm^c_{v,\text{CPU}}$$  \hspace{1cm} (4.6)

Equations 4.7 - 4.8 shows the placement constraints, they assure the correct placement of virtual machines on their designated physical machines. Equation 4.7 states that each virtual machine has to be and can only be served by one physical machine. Let $DH(v)$ be the potential destination hosts for a virtual machine $v$. Equation 4.8 ensures that a virtual machine can only be hosted by
one of its potential destination hosts. The selection of the potential destination hosts for a virtual machine $v$ is governed by the actual and predicted resources required by the virtual machine, the availability of these resources on the host, the host resource prediction (i.e. if the host is predicted to be overloaded or under-loaded), and the migration cost from the source to the destination node.

\[
\sum_{p \in P} H(p, v) = 1, \quad \forall v \in VM
\] (4.7)

\[
\sum_{p \in DH(v)} H(p, v) = 1, \sum_{p \in P \setminus DH(v)} H(p, v) = 0, \quad \forall v \in VM
\] (4.8)

Equations 4.9-4.10 introduce the actual and predicted resources utilization constraints. Since virtual machines deployed on a physical machine require some amount of resources, these resources should not exceed a specific utilization level from the resources offered by the physical machines (unless indicated, we consider the utilization threshold = 70%). In this formulation, $c$ is the resource type. Note that all types of resources are considered (CPU, memory, network, and disk).

\[
\sum_{v \in VM} (H(p, v) \times vm^c_v) \leq U^c_p, \quad \forall p \in P
\] (4.9)

\[
\sum_{v \in VM} (H(p, v) \times \hat{vm}^c_v) \leq U^c_p, \quad \forall p \in P
\] (4.10)

Equation 4.11 calculates the power consumed by a switch. Which is the power consumed by its fixed components $\epsilon^{Sactive}$, such as chassis, fans, line cards, ... etc., in addition to the power consumed by each active port $\epsilon^{Port}$.

\[
Spow(s) = \epsilon^{Sactive} \cdot On(s) + \sum_{n \in N_i} \epsilon^{Port} \cdot On(s, n)
\] (4.11)

Equations 4.12-4.13 present the flow constraints. Equation 4.12 states that a flow should always starts/ends at the host that contains the source/destination virtual machine. Equation 4.13 ensure that the virtual machines will use local bus if they were placed on the same server, otherwise the transmission should start at the server that hosts the source $vm$ and ends at the server that hosts the destination $vm$. 
\[ \sum_{s \in S} (\text{route}(f, (p, s))) \leq H(p, \text{src}(f)), \sum_{s \in S} (\text{route}(f, (s, p))) \leq H(p, \text{dest}(f)), \forall p \in P, \forall f \in F \] (4.12)

\[ H(p, \text{src}(f)) - H(p, \text{dest}(f)) = \sum_{s \in S} (\text{route}(f, (p, s))) - \sum_{s \in S} (\text{route}(f, (s, p))), \forall p \in P, \forall f \in F \] (4.13)

Equations 4.14-4.17 show the links constraints. Equation 4.14 introduces the active link constraint. It states that an active link connects two active switches or a switch and a server.

\[ \text{On}(i, j) \leq \text{On}(i), \text{On}(i, j) \leq \text{On}(j), \forall i, j \in E, \forall i \in S \] (4.14)

Equation 4.15 states the bidirectional link power constraint which means both directions of a link \((i, j)\) should have the same on/off power status. Likewise, equation 4.16 ensures that for every active link \(\text{On}(i, j) = 1\), both directions have the same capacity limits \(C_{i,j}\).

\[ \text{On}(i, j) = \text{On}(j, i), \forall i, j \in E \] (4.15)

\[ \text{On}(i, j) \cdot C_{i,j} = \text{On}(i, j) \cdot C_{j,i}, \forall i, j \in E \] (4.16)

Equation 4.17 introduces the satisfiability constraint. It shows that the summation of all traffic flow loads passing through link \((i, j)\) is always less than or equal to the capacity limit of that link \(C_{i,j}\).

\[ \sum_{f \in F} (\text{route}(f, (i, j)) \cdot bw(f)) \leq \text{On}(i, j) \cdot C_{i,j}, \forall i, j \in E \] (4.17)

Equation 4.18 shows the active switch constraint. Let \(N_i \in N\) be the set of ports in a switch and \(|N_i|\) is the cardinality of \(N_i\), then equation 4.18 ensures that a switch will be turned off only if all its ports are turned off.

\[ |N_i| \cdot (1 - \text{On}(i)) \leq \sum_{j \in N_i} (1 - \text{On}(i, j)), \forall i, j \in E, \forall i \in S \] (4.18)
Equations 4.19-4.20 present utilization constraints. Equation 4.19 calculates the link utilization $u$ for each link. Where link utilization is the summation of every traffic flow load passing link $(i,j)$ to the capacity of that link. Equation 4.20 ensures that the utilization of every link is always less than or equal to a predefined upper link utilization threshold $U_{upper}$ (unless indicated, we consider $U_{upper} = 0.80$).

$$u_{i,j} = \sum_{f \in F} \left( \frac{\text{route}(f,(i,j)) \cdot bw(f)}{C_{i,j}} \right), \forall i,j \in E$$

(4.19)

$$u_{i,j} \leq U_{upper}, \forall i,j \in E$$

(4.20)

Since mixed integer linear programming is NP-hard, the proposed formulation is not practical for large data centers.

### 4.7 Heuristic Algorithm

To overcome the exponential increase in the optimization software computation time, a heuristic algorithm solving the data center energy-saving problem was developed. In data center environment, traffic demands fluctuate frequently. For that reason, heuristic algorithm is preferred to solve our optimization model in real time. The algorithm takes input similar to the MILP and is divided into two main stages: virtual machine placement and consolidation and network flow routing and consolidation.

The virtual machine placement and consolidation stage consists of two parts: initial virtual machine placement and dynamic virtual machine consolidation. The initial virtual machine placement stage targets finding a virtual machine placement on the hosts such that no resource violation occurs. Since no historical data available, no virtual machine nor host prediction can occur. The algorithm places the virtual machine using First Fit Decreasing (FFD) method, which is one of the most efficient algorithms to solve the bin-packing problem [40]. However, due to workload fluctuations, resource demands changes over time. So, the initial placement will not be efficient anymore and there is a need for a virtual machines consolidation to update the current placement and provide a more optimized solution which is presented in the second stage.
Algorithm 4 and 5 show the pseudocode of the proposed virtual machine consolidation for both overloaded and underloaded hosts. The algorithm runs periodically to solve any resource utilization constraint violation (Algorithm 4) and saving more energy by migrating \textit{vms} from under utilized hosts to put them into standby mode.

The algorithm starts by categorizing the active hosts into three sets: OverloadedHosts, UnderloadedHost, and NormalHosts. The OverloadedHosts set consists of all hosts that violates one of its resource threshold constraint. UnderloadedHost is the set of hosts that none of its resource utilization exceeds 10% from the past execution. NormalHosts is the set of active hosts that are not overloaded nor underloaded and is not predicted to become overloaded in the near future. Algorithm 4 deals with each of the overloaded hosts by sorting their virtual machines in descending order of their total utilized resources. Starting with the virtual machine with the highest total utilized resources, the algorithm try’s to find a target host from the set of normal hosts. To reduce the search space within the set of normal hosts, the virtual machine should be migrated to a normal host within its rack or pod. Furthermore, it should be assured that the migration will not cause the target server to become overloaded or predicted overloaded. If found, the virtual machine will be migrated to the target host. if the migration of the virtual machine solves the host resource violation (i.e. the host not overload any more), the host will be removed from the overloadHost list and it will be added to one of the other two lists. If the migration did not solve the resource violation, the process will be repeated with the second highest virtual machine and so on. if no more hosts in the normal hosts set able to handle the overloaded hosts virtual machines, the same procedure will be done using the set of underloadhosts. Finally, if no normal nor underloaded hosts able to handle hosting virtual machines from overloaded hosts, a new host will be turned on and the virtual machines will migrate to it.

Algorithm 5 aims to save more energy by migrating the virtual machines from underloaded hosts to put them into standby mode. Similar to the algorithm 4, it categorizes the hosts into the same three categories. Then, it sorts the hosts in ascending order of their resource utilization. For the host with the lowest required resource, the algorithm tries to find a target host or a set
Algorithm 4 VM Consolidation - Overloaded Hosts

1: Input: \( P, VM, H_{pred}, VMpred, res(P), res(VM), On(p) \)
2: \( OverloadHosts = res(P) \mid "overloaded" \)
3: \( UnderloadHosts = res(P) \mid "underloaded" \)
4: \( NormalHosts = res(P) \mid "normal" \)
5: if \( OverloadHosts = \emptyset \) then
6: break
7: for \( p \in OverloadHosts \) do
8: Sort \( vm \in p \) in descending order based on total utilized resources
9: for \( vm \in p \) do
10: if \( \text{FindTargetHost}(vm, Normal) \) then
11: Migrate \( vm \)
12: if \( p \notin OverloadHosts \) then
13: \( OverloadHosts = OverloadHosts - p \)
14: next \( p \)
15: for \( vm \in p \) do
16: if \( \text{FindTargetHost}(vm, Underload) \) then
17: Migrate \( vm \)
18: if \( p \notin OverloadHosts \) then
19: \( OverloadHosts = OverloadHosts - p \)
20: next \( p \)
21: if \( \exists p_{new} \in standbymode \) then
22: Turn on \( p_{new} \)
23: Migrate \( vm \) to \( p_{new} \)
of hosts that can satisfy hosts virtual machines. The target host(s) should be in within the source host rack or pod to reduce the migration cost on the network, it’s in the set of normal hosts, and the migration will not move the target host to be overloaded or predicted overloaded. If a target host(s) is found, all vms from the source host will be migrated and the host will be set into standby mode. If there is no more normal host available and there are more than 1 host in the set of UnderloadHosts, the algorithm searches for a target hosts within UnderloadHosts set. The migration conditions are similar to migration to normal hosts. The computational cost for this algorithm is $O(P.VM^2.\log VM)$.

Algorithm 5 VM Consolidation - Underloaded Hosts

1: **Input:** $P, VM, Hpred, VMpred, res(P), res(VM), On(p)$
2: $OverloadHosts = res(P) | "overloaded"$
3: $UnderloadHosts = res(P) | "underloaded"$
4: $NormalHosts = res(P) | "normal"$
5: If $UnderloadHosts = \phi$ then
6: break
7: $\forall p \in UnderloadHosts$ Sort them in ascending order of their total utilized resources
8: for lowest $p \in UnderloadHosts$ do
9: if $FindTargetHosts(p, Normal) \neq false$ then
10: Migrate all $vms \in p$
11: Put $p$ into standby mode
12: else
13: Continue
14: if $|UnderloadHosts| > 1$ then
15: for lowest $p \in UnderloadHosts$ do
16: $FindTargetHosts(p, UnderLoad)$

For the network flow routing and consolidation stage we use our previous work [88], which is an abstract model that saves data center network energy while maintaining the network performance from traffic surges. we proposed a a light-weight heuristic algorithm that combines setting-up safety margin threshold and load balancing technique together is presented to save energy and maintain network performance to handle traffic surges.

The heuristic algorithm starts by setting up predefined safety thresholds on each link capacity. Then, it continuously monitors the utilization of network links and balances the loads on active
links using Valiant Load Balancing (VLB) mechanism [17]. A decision to turn on new switches or links can be taken if these thresholds are exceeded. Using this algorithm, the safety margins and the load balancing mechanism allow the network to handle traffic surges, while maintaining its performance. On the other hand, switches grouping and links consolidation will also take place if the loads on the networks switches and links are under-utilized. This will allow turning off some active ports and switches to lower network power consumption.

4.8 Performance Evaluation

This section presents the evaluation of our proposed heuristic algorithm. The evaluation is conducted to show that the algorithm achieves a considerable amount of energy saving while maintaining performance.

We compared the proposed heuristic to other algorithms including Most Full First (MFF) and Least Full First (LFF) bin packing algorithms for virtual machine placement. Most Full First will assign the virtual machine to the most full host that can satisfy the virtual machine resource demands. On the other hand, Least Full First assign the virtual machine to the least full host. No virtual machine migration is implemented on these methods. Another method [84] that uses virtual machine consolidation to save energy and minimize the cost of live migration on the network. This method does not provide energy saving for network devices so we will refer to it as no network.

4.8.1 Simulation Setup

The proposed heuristic was implemented in CloudSimSDN [89]. It is an extension of the popular CloudSim simulator [90] that supports different software defined networks features. We consider a fat-tree data center with \( k = 8 \). The data center includes a 16 core switches, 32 aggregate switches, 32 access switches, and 128 hosts. Each host is equipped with 8 core CPU. Figure 4.2 shows Fat tree network topology with \( k = 4 \).

Cloud data center workloads fluctuate frequently. In this evaluation, we investigate the data center traces provided by Wikipedia data center, which is available for public. Specifically, we
investigate the statistical Page view data for selected Wikimedia projects. For each hour, the traces consists of the page view count and the amount of bytes transmitted as a respond. Figure 4.3 and 4.4 shows the how number of requests and the bytes transmitted as a response varies each hour. For the experiments that include migration, the monitoring interval is set to 2 minutes to collect the utilization of VMs, hosts, flows, and links. The migration is attempted every 20 minutes. The overall simulation time is 2 hours.
4.8.2 Simulation results

In this section we present the simulation results to compare the proposed algorithm against MFF, LFF, and no network in terms of servers, switches, and total energy consumption and network time. Figure 4.5 shows the energy consumed by the servers. It can be seen that the proposed algorithm uses least energy compared to the other method. This is due to the quality of virtual machine consolidation process. The proposed algorithm uses hosts and virtual machines resource prediction which provide more information for better placement of the virtual machines. The no network uses only the current status of the resource utilization without any prediction. Both MFF and LFF do have any virtual machines migration, thus, it can’t cop with workload fluctuations. LFF spread the workload over all hosts resulting in the worst energy consumption. MFF uses the best fit initially so it saves energy. But as the workloads varies over time, MFF can’t cop with this variation and misses opportunities for more energy saving.

Figure 4.6 shows the energy consumed by switches. The results shows that the proposed algorithm outperforms all other algorithms. The proposed algorithm uses flow consolidation to move the flows into subset of the network devices and turn off unused ones. The proposed algorithm consumes 0.534 KW.h compared to 0.9557, 0.7836, and 1.185 for the no network, MFF, and LFF, respectively. Likewise, figure 4.7 show the total energy consumed by the data center.
Finally, we compare the proposed algorithm against other algorithms in terms on average response time. Figure 4.8 shows the average response time for all algorithms. The results states that the proposed algorithm has the best average response time. Specifically, the proposed algorithm has average response time of 1.248 seconds compared to 1.694, 2.448, and 2.9296 for no network, MFF, and LFF, respectively.
The architectural design of data centers can be exploited for energy saving. Most research on literature focuses on optimizing energy saving for the servers and network separately. In this paper, we propose a joint optimization for minimizing the server-side and network side of the data center. The optimization is part of a framework that collect and predict servers and virtual machines resource utilization which will be used as an input for the joint optimization. The results of the joint optimization includes the new placement of virtual machines. For practical implementation on large scale data centers, a two stage heuristic algorithm is proposed. The heuristic first find near
optimal solution for the server side then it uses an abstract performance model for saving network energy. The proposed algorithm was evaluated using CloudsimSDN with real Wikipedia traces. The results show that proposed algorithm saves servers and network energy while maintaining performance.
CHAPTER 5. CONCLUSIONS AND FUTURE WORKS

In this thesis, we present our proposed techniques on: 1. how to achieve considerable amount of energy saving in cloud data centers; 2. how to maintain data center performance, 3. how to provide a practical and scalable solutions that can be implemented in modern enterprise data centers. Chapter 2 presents network-side energy saving for cloud data centers. The idea is to move traffic flows into a subset of the network and turn off unused switches. The proposed solution uses valiant load balancing technique and link threshold to maintain network performance and handle traffic surges. Chapter 3 introduces a weighted sum multi-objective optimization for saving servers’ energy and minimizing the effect of live migration on network links. Ch 4 combines servers and switches energy saving in a joint optimization to maximize data center energy saving. The proposed solution deals with the problems of network and server energy saving combined. We summarize the main contributions and proposed methods discussed in this thesis:

- We propose a technique to save data center energy while preserving network performance from traffic surges.

- For large scale data centers, we design a heuristic algorithm that sets safety thresholds on link capacities and uses valiant load balancing technique on active links. The proposed heuristic is abstract and can be applied to any switch-centric topology in similar fashion.

- We implement the proposed heuristic algorithm using GreenCloud simulator and compared it with other algorithms. Both synthetic and real traces demonstrate that the heuristic algorithm saves considerable amount of energy with minimum effect on the DCN.

- We propose a dynamic virtual machine framework with the objective to minimize energy consumption and virtual machines migration effect on the network.
We propose the Average Imbalance Score metric for both switches and links to evaluate the performance of the load balancing mechanism. Using this metric, we show that the heuristic algorithm improves the imbalance score for links and switches by more than 50.

A multi-objective optimization formulation for server-side energy saving and time to migrate virtual machines is introduced. The optimization considers all servers resources in its solution (CPU, memory, network, and disk) such that energy wastage and performance bottlenecks caused by resource wastage are eliminated.

We present a two-stage greedy heuristic algorithm that achieves near-optimal energy saving and low computational complexity. This heuristic is practical solution for large size data centers.

We build a real testbed data center to evaluate our proposed heuristic. We choose workloads from web service applications, big data benchmarks, i.e., HiBench to Docker software containers that represent today’s cloud computing environment.

We propose server and network joint optimization to maximize data center energy saving while maintaining performance.

A framework that monitors the status of the data center is introduced. The framework collects and predicts servers resource utilizations and current network traffic. The joint optimization will take collected data as an input and provide an optimal solution for saving energy. Live migration commands will take place based on the optimal solution and unused server and switches will turn into standby mode or turn off.

A two-stage heuristic algorithm is presented for practical implementation on large scale data centers.

We implement the proposed heuristic using CloudSimSDN simulator and compare it to other algorithms. Real Traces collected from Wikipedia are used to show that the proposed heuristic can achieve considerable amount of energy while maintaining performance.
For future works on data center energy saving, a complete framework that maximizes energy saving can be achieved by applying Dynamic Voltage and Frequency scaling (DVFS) method to data center servers along with network-side and server-side energy saving. A more complete framework is to also include cooling to the joint optimization while applying DVFS to servers. Such practical framework will maximize the saved energy and decrease the operational costs for cloud providers.


[34] Cisco press, “Cisco data center infrastructure 2.5 design guide.”


