electronics

MDPI

Article

An Adaptive Routing Framework for Efficient Power Consumption in Software-Defined Datacenter Networks

Mohammed Nsaif [1], Gergely Kovásznai [2], Anett Rácz [3], Ali Malik [4] and Ruairí de Fréin [4,*]

[1] Department of Information Technology, University of Debrecen, 4032 Debrecen, Hungary; mohammed.nsaif@mailbox.unideb.hu
[2] Department of Computational Science, Eszterhazy Karoly Catholic University, 3300 Eger, Hungary; kovasznai.gergely@uni-eszterhazy.hu
[3] Department of Applied Mathematics & Probability Theory, University of Debrecen, 4032 Debrecen, Hungary; racz.anett@inf.unideb.hu
[4] School of Electrical and Electronic Engineering, Technological University Dublin, D07 EWV4 Dublin, Ireland; ali.malik@tudublin.ie
* Correspondence: ruairi.defrein@tudublin.ie

Abstract: Data Center Networks (DCNs) form the backbone of many Internet applications and services that have become necessary in daily life. Energy consumption causes both economic and environmental issues. It is reported that 10% of global energy consumption is due to ICT and network usage. Computer networking equipment is designed to accommodate network traffic; however, the level of use of the equipment is not necessarily proportional to the power consumed by it. For example, DCNs do not always run at full capacity yet the fact that they are supporting a lighter load is not mirrored by a reduction in energy consumption. DCNs have been shown to unnecessarily over-consume energy when they are not fully loaded. In this paper, we propose a new framework that reduces power consumption in software-defined DCNs. The proposed approach is composed of a new Integer Programming model and a heuristic link utility-based algorithm that strikes a balance between energy consumption and performance. We evaluate the proposed framework using an experimental platform, which consists of an optimization tool called LinGo for solving convex and non-convex optimization problems, the POX controller and the Mininet network emulator. Compared with the state-of-the-art approach, the equal cost multi-path algorithm, the results show that the proposed method reduces the power consumption by up to 10% when the network is experiencing a high traffic load and 63.3% when the traffic load is low. Based on these results, we outline how machine learning approaches could be used to further improve our approach in future work.

Keywords: DCN; integer programming; optimization; power consumption; QoS; SDN

Citation: Nsaif, M.; Kovásznai, G.; Rácz, A.; Malik, A.; de Fréin, R. An Adaptive Routing Framework for Efficient Power Consumption in Software-Defined Datacenter Networks. *Electronics* **2021**, *10*, 3027. https://doi.org/10.3390/electronics10233027

Academic Editors: Dongkyun Kim, Qinghe Du, Mehdi Sookhak, Lei Shu, Nurul I. Sarkar, Jemal H. Abawajy and Francisco Falcone

Received: 18 October 2021
Accepted: 30 November 2021
Published: 4 December 2021

1. Introduction

In recent years, energy consumption has become an important issue in a range of technology sectors such as Wireless Sensor Networks (WSN), Mobile Crowd Sensing (MSC), Internet of things (IoT) and Data Center Networks (DCN) [1–3]. Data centers aim to provide reliable and scalable computing infrastructure for massive Internet services [4]. Data centers promise several benefits such as: (1) flexibility without sacrificing forwarding performance; (2) high efficiency which is achieved by optimizing routing; (3) ease of deployment/administration; and finally, (4) cost reduction [5]. The recent Software-Defined Networking (SDN) paradigm decouples the network control plane from the data plane, as a result, the level of network programmability is increased. New SDN architectures make new, bespoke solutions for a wide range of networking issues possible. In terms of cost reduction in DCNs, two main strategies have been adopted: (1) smart energy management and (2) software-defined power consumption reduction. These approaches are similar. They have the same goal, which is to enhance the energy consumption efficiency by

reducing redundancy and capital costs. According to the Energy Efficiency Status Report which was published by the European Commission Joint Research Center in 2012 in [6], the total energy consumption of data centers in 2007 was estimated to be 56 TWh. This constitutes 2% of the total energy consumption in the EU each year. It is expected that this power consumption will nearly double to reach 104 TWh/year in the future [6,7]. In the same context, in 2017, other studies showed that the energy consumption of DCNs was estimated to account for 1.4% of the global electricity energy usage and the Compound Annual Growth Rate (CAGR) was estimated to be 4.4% for the period 2007–2012. This is much higher than the projected 2.1% increase in global demand from 2012 to 2040 estimated in [8]. In Table 1, we summarize the study conducted in [6], which shows the decrease in energy consumption of DCNs achieved in Europe due to consolidation, virtualization and energy-efficient technologies especially in cooling over the period 2013 to 2020.

Table 1. Decreasing energy consumption in DCNs.

Western Europe	2013	2014	2015	2016	2017	2018	2019	2020
Net data center space (thousands of m^2)	10,256	10,221	10,105	10,055	9875	9555	9365	9155
Average power density (kW/m^2)	1.1	1.1	1.2	1.2	1.3	1.3	1.2	1.3
Total power usage (GW)	11.3	11.2	12.1	12	12.8	12.4	11.3	10.9

It has been suggested that there are three different strategies that can be pursued to improve the energy power consumption of DCNs. The first approach consists of improving efficiency by being aware of the energy consumption associated with different types of traffic. This approach is inspired by the fact that network components are frequently underutilized. The key concept of this technique is to turn network components on and off based on traffic demand. It is projected that this strategy has the ability to save up to 50% of the total energy usage when traffic loads are low (for example, during the night) in the following studies [4,9–11]. The second approach focuses on making end-systems aware of possible energy savings. This class of approaches attempts to run underutilized physical server tasks on a fewer number of servers in SDN-based DCNs. Notable proponents of this approach include the authors of [12–15]. The third class of approaches is a rules placement technique. In this approach the central controller converts high-level policies into switch-readable rules. On the one hand, rules placement is considered to be an NP-hard problem and therefore, heuristic solutions are required. On the other hand, heuristic-based methods can not guarantee that the optimal solution will be found. There is an inherent trade-off between obtaining the optimal solution and having a solver which is practical, e.g., it runs in a short enough time to yield a solution which is useful in an engineering setting. In many cases, the complexity that results from the constraints required to obtain the optimum solution causes the performance of these solvers to be unpractical [15–18]. In recent years, Internet traffic volume has been growing exponentially due to the dramatic increase in the use of live video streaming, video games and social networks [19]. Managing network components by considering the footprint of traffic in DCNs has the potential for large energy consumption savings.

In this paper, we adopt a traffic-aware approach to reduce the power consumption in DCNs. We propose a new energy-efficient adaptive approach, which is called the Fill Preferred Link First (FPLF) algorithm. FPLF aims to maintain the QoS and the energy consumption of DCNs. This is achieved by continuously monitoring the traffic conditions of the DCN by utilizing the OpenFlow protocol [20] to obtain the topology state and data traffic information. It picks the most energy-efficient path that is below a pre-defined threshold value. We demonstrate that FPLF can maintain satisfactory QoS whilst reducing power consumption in DCNs.

This paper is organized as follows. Section 2 introduces various power consumption reduction techniques from the literature. The problem statement is given in Section 3. The proposed model is presented in Section 4. The proposed algorithm is presented in Section 5. The implementation and and an analysis of the performance of the FPLF algorithm is presented in Section 6. We conclude by summarizing the key contributions and results in Section 7 and outline some future directions.

2. Related Work

This section presents important, recent studies which have adopted Integer Programming (IntP) to address the power consumption reduction challenge for DCN routing algorithms. We outline the strengths and limitations of current approaches. We start by considering DCN routing algorithms in enterprise and data center environments. The state-of-the-art forwarding approach uses Equal Cost Multipath (ECMP) to stripe flows across available paths using flow hashing statically [21]. Collisions overwhelm switch buffers and degrade the overall switch used. Due to this, static mapping of flows to paths does not account for the current network use or the size of the flows. To resolve this problem, Al-Fares et al. proposed an extensible and dynamic flow scheduling system called Hedera in [22]. Hedera is a system that exploits path diversity in DCN topologies to enable a near-ideal bisection bandwidth for a range of traffic patterns. Hedera collects flow information from constituent switches, computes non-conflicting paths for flows, and instructs switches to re-route traffic accordingly. Its goal is to maximize the aggregate network utilization bisection bandwidth. However, this approach impacts active flows by changing the paths as well as increasing power consumption levels.

The Elastic Tree algorithm was proposed by Heller et al. in [4]. This approach can dynamically change idle network devices to sleep mode to save energy. The study suggests that a topology-aware heuristic optimizer should split the flow and that it finds the link subset easily. ElasticTree's experimental results reveal that it can achieve an energy saving of up to 50%. One drawback is that it does not consider the correlation among the flows. In light of this observation we add this type of constraint in our enhanced version of the model. The two-directions study presented by Luo et al. in [23] was called FLOWP. FLOWP attempts to achieve both power reduction and QoS. An IntP formulation for power-efficient flow scheduling was proposed. In the formulation, the network status and the minimum threshold for the efficiency of links and switches was considered. The result proved a saving of 30% was achievable and that the QoS was improved compared to the approach in [4]. Traffic-aware energy approaches for cloud computing infrastructure are introduced in [24], where two components are suggested: (1) The Data Manager (DM) to control the states of switches and to detect input traffic; (2) Power Manager (PM) to control the operating system modes of switches . The results reveal that all switches save up to 30–35% on average. The Next Shortest Path and Next Maximum Utility heuristics prioritized performance and energy savings, respectively. A single heuristic proposed by Assefa and Ozkasap in [25] maximized a parameter called the Energy Profit Threshold (EPT) to achieve both energy-saving and performance at the same time. Experiments using a real network architecture and traces provided by SNDLib [26] indicated that a 50% energy saving was possible. In the same context, the RESDN was also presented by Assefa and Ozkasap in [27] as an energy efficiency metric that quantified energy efficiency based on link utility intervals, the IntP formulation, and the method for maximizing the ratio for energy saving of the SDN network. A Genetic Algorithm-based heuristic was presented in [13] to minimize the number of modified rules and the network power consumption in flow tables. The authors' aim was to minimize the cost of re-configuration in the flow tables when traffic volume changed. The results showed that the rule size reduction contributed to reducing power consumption by approximately 20%, compared to similar methods.

The approaches considered above look to deliver high QoS whilst achieving energy efficiency improvements. There exist methods that consider QoS from the perspective of link failure probability and the frequency of usage of links [28] but that do not consider

energy efficiency. Finally, the authors in [1] introduced fast heuristic methods as well as iterative heuristic algorithms to select optimal controller placement and switch assignment to minimize total control traffic in SDN-based IoT networks with the goal of reducing power consumption. The study revealed that the algorithms achieved the near-optimal solution within a shorter computation time. However, it is possible that the framework for encoding prior information into path allocation could be adapted to consider the QoS-energy efficiency trade-off considered here.

Many of the above studies used Linear Programming and IntP to solve the problems and to give reasonable solutions which were scalable. However, some open problems need to be further investigated such as the ability to include other network operation conditions. For instance, traffic-aware methods should be able to support scalability as the traffic load gets larger. In this case the complexity of finding feasible solutions should also be considered. To address these challenges, this paper suggests an enhanced model that introduces an acceptable trade-off level between energy savings and network performance. This trade-off is suggested by the FPLF algorithm based on link utilization classification. A more detailed description of the approach is given in Section 5, but first, we define the problem and outline how the proposed solution operates.

3. Problem Statement and Proposed Solutions

For the majority of the time, it has been reported that the DCN is in a low-load state [5]. The problem is that energy consumption of DCN resources, such as links, is wasteful when the utilization level is low. This is because engaging certain equipment in DCN activities, e.g., routing, does not justify the additional cost of the associated energy consumption [4]. Therefore, our proposed approach causes a gradual increase in the number of active links as the traffic demand increases. The opposite strategy is pursued when the traffic demand is low.

Section 5 clarifies how the FPLF algorithm addresses this problem. In summary, the FPLP algorithm aims to minimize the level of power consumption subject to achieving an acceptable level of QoS. We continue by highlighting the general concept of the problem. A study which measured and modeled the power consumption of OpenFlow switches was published in 2014 [29]. The study summarized the sources of energy consumption in SDN, and one of those sources was active ports. According to the authors, the under-utilization of links caused an increase in energy consumption due to an increase in the power consumption of ports. This is because the underlying technology used to establish a link is two ports between two switches. We conclude that minimizing the number of under-utilized links is one of the main factors that can be exploited to save power. Thus, the mission of FPLF is to maximize the benefit from using the set of active links and their capacity, i.e., to maximize the utilization of links.

We illustrate our approach in more detail in Figure 1. We construct a topology as an example in order to analyze the performance of the two forwarding methods we propose below. There are 20 switches and 16 hosts. Assume the links have the same bandwidth, BW. The source and destination hosts of data flows f_1 and f_2 are C, H, and D, G, respectively. The volume of traffic injected by f_1 and f_2 are V_1 and V_2, respectively. Assume that the flows are sent to the DCN simultaneously at $t = 0$. When flows arrive at 16, the FPLF algorithm checks the link utilization matrix so that it fills the active links to 90% of their maximum utility. This percentage utilization is targeted for performance reasons between the source and the destination. Once it is exceeded we open new paths. The end result is that we keep as many links as possible in the off state. From a routing perspective, the path $16 \rightarrow 13 \rightarrow 2 \rightarrow 9 \rightarrow 12$ is shared between both of f_1 and f_2 if and only if the utility of the links is under the threshold value in the link utilization matrix.

At time $t = 5$, an additional flow is injected into the network. The host A sends the flow f_3 to the destination H. Assume that the total traffic volume of the flows exceeds the 90% target utilization, $\sum_{i=1}^{3} V_i > 0.9$, when f_3 arrives at A_1 FPLF checks the link utilization matrix again, and directly opens a new path $15 \rightarrow 13 \rightarrow 1 \rightarrow 9 \rightarrow 11 \rightarrow 19 \rightarrow 12$ in response.

Note that this new path has more hops. In summary, FPLF classifies all the links in the first path as being overloaded links. As a result of this overloaded status, FPLF opens a new path and classifies the new path's links as being under-utilized links. All the other links are classified as being idle links, i.e., unused links.

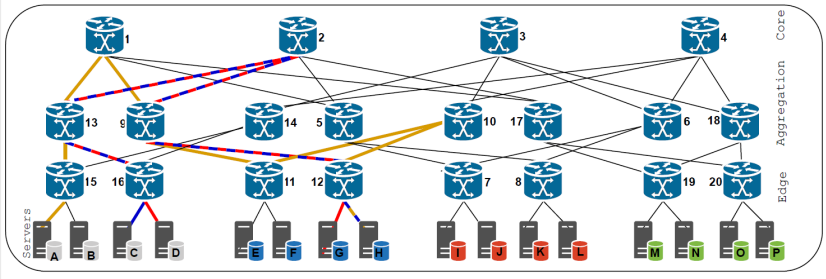


Figure 1. Motivational example of the FPLF algorithm's behavior.

4. Proposed Model

We start first by outlining some of the notation that will be used in the rest of the paper. The notation is summarized in Table 2.

Table 2. List of notation.

Parameter	Description
$\mathbb{S}$	Set of nodes (switches) in the DCN topology, $\mathbb{S} = \{S_1, \ldots, S_n\}$
$\mathbb{E}$	Set of edges where $e_{ij} \in \mathbb{E}$ represents the connection between two switches S_i and S_j
BW_{ij}	adjacency matrix scaled by the bandwidth of the edges $\mathbb{E}$ equal to 1 Mbps for all links
L_{ij}	$= \begin{cases} 1, & \text{if } e_{ij} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$
$\mathbb{F}$	Set of flows
$f = (f.S_r, f.D_s, \lambda_f)$	A flow f $\in \mathbb{F}$ represented by source, destination and packet rate
$FR(f,i,j)$	$= \begin{cases} 1, & \text{if flow } f \text{ passes through edge } e_{ij} \\ 0, & \text{otherwise} \end{cases}$
$\mathbf{U}$	Utilization matrix where U_{ij} represents the utilization of the link e_{ij}
T	Input traffic matrix where T_{ij} denotes input traffic of e_{ij}
$\mathbf{C}$	Link-cost matrix where C_{ij} denotes the cost of the link e_{ij}
k	number of the pods in the fat-tree topology
ESP	energy-saving path
s_r	source switch
d_s	destination switch

4.1. Network Model

The network topology is modelled as an undirected weighted graph G $= (\mathbb{S}, \mathbb{E})$, which has a vertex-set, $\mathbb{S}$, and an edge-set, $\mathbb{E}$. Each node is an OpenFlow switch and the i-th switch is denoted S_i. The role of each switch is to forward information based on the path selected by the network controller. Each edge in the graph is a link. The link between i-th and j-th switches is denoted by e_{ij}. Network links can be in either ON or OFF mode. We define the binary variables: L_{ij} to confirm the current mode state. The value of L_{ij} is 1 if the link between switches i and j is active. In other words, the link can transmit packets between two ports. The value of L_{ij} is 0 otherwise. In order to capture the flows traversing the topology, the variable $FR(f,i,j)$ takes the value 1, which means that f traverses the edge e_{ij}. If e_{ij} is not traversed by the flow $FR(f,i,j)$, it is set to 0. The link utilization U_{ij} represents the ratio of the size of the flows passing through the edge e_{ij} and the link bandwidth BW_{ij}. When this ratio is scaled by 100, the utility of a link ranges from 0 to 100%. A link utilization matrix, $\mathbf{U}$, is constructed by considering the utilization of each edge. The traffic of the DCN is represented by the set of flows $\mathbb{F}$, where each $f \in \mathbb{F}$ is defined as $f = \left(f.S_r, f.D_s, \lambda_f\right)$. A flow represents a group of packets which have the same source and destination address, that travel along the same route to reach their destination. The source

and destination switches are denoted by s_r and d_s, respectively. Finally, the packet rate of flow f is denoted λ_f and is measured in bits per second.

4.2. Optimization Model

In this model, we consider the links as the main energy-saving components in the network. To accommodate the provided traffic, the model uses the set of active links which has the smallest cardinality. The optimization model considers the following problem characteristics: (1) the optimization model's parameters refer to a snapshot of the network state for the sake of simplicity. This means that the model considers the network state at a specific moment of time; (2) the model starts with a standard multi-commodity flow problem. The constraints include flow conservation, link capacity, demand satisfaction, and the total number of active links; and finally, (3) splitting a single flow into packets across multiple links in the topology could save the energy by increasing overall link utilization. However, the reordered packets at the destination, due to varied path delays, can degrade the performance. As a result, we incorporate restrictions into our formulation based on the entire flow.

We assume that all the links have the same energy consumption at the beginning. The objective function is defined as

$$\text{minimize}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} L_{ij}\right). \tag{1}$$

The objective function in Equation (1) minimizes the number of active links, which reduces the energy consumption in turn. We set $L_{ij} = 0$, when there is no physically connection between S_i and S_j. When a physical link does exist, the value of the link is either 0 or 1 depending on the traffic demand. The model also takes into account the correlation between the links and traffic volume. Therefore, the first and second constraints, which are defined in Equations (2) and (3), respectively, define the relationship between the traffic volume and link state,

$$\frac{T_{ij}}{BW_{ij}} \leqslant L_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \text{ and} \tag{2}$$

$$U_{ij} = \sum_{\forall f \in \mathbb{F}} FR(f,i,j) \cdot \lambda_f, \quad \forall e_{ij} \in \mathbb{E}. \tag{3}$$

The third constraint is defined in Equation (4). It states that the utility of each link should be less than the link capacity BW^{th} minus the current flows described by the input traffic matrix. This constraint acts to bound the value of U_{ij}, by considering the active flows f_{ij} and their packet rate values λ_f ,

$$U_{ij} \leqslant BW^{th} - \mathrm{T}_{ij}, \quad \forall e_{ij} \in \mathbb{E}. \tag{4}$$

The fourth and fifth constraints in Equations (5) and (6) guarantee the path of each flow goes from a source to a destination,

$$\sum_{i=1}^{n} FR(f, f.S_r, i) = 1, \quad \forall f \in \mathbb{F}, \tag{5}$$

$$\sum_{i=1}^{n} FR(f, i, f.D_r) = 1, \quad \forall f \in \mathbb{F}. \tag{6}$$

To avoid packet loss, the sixth constraint in Equation (7) states that the incoming and outgoing flows of the transit switches, which are neither sources nor destinations, should be equal,

$$\sum_{\substack{i=1 \\ i \neq \mathrm{s_r}(f)}}^{n} FR(f,i,j) = \sum_{\substack{i=1 \\ i \neq \mathrm{d_s}(f)}}^{n} FR(f,j,i), \quad \forall f \in \mathbb{F},\ j \in \mathbb{S}. \tag{7}$$

To avoid the looping between nodes, we formulate the seventh constraint as follows in Equation (8),

$$FR(f,i,j) + FR(f,j,i) \leqslant 1, \quad \forall f \in \mathbb{F},\ i,j \in \mathbb{S}. \tag{8}$$

Finally, the eighth constraint in Equation (9) illustrates the correlation between links and flow status variables, by pointing out that the link should be active if there is at least one flow that passes through it,

$$FR(f,i,j) \leqslant L(i,j), \quad \forall f \in \mathbb{F},\ i,j \in \mathbb{S}. \tag{9}$$

4.3. IntP formulation

Our goal is to obtain the optimal solution, which performs appropriate link and flow assignments, so that the traffic demands are satisfied. However, determining the best flow assignment is an NP-complete problem according to the analysis in [4].

We implement our proposed model in LinGo [30], which is a high-level language for optimization modeling. The inputs to the model are summarized by the notation in Table 2. They include: the topology (fat-tree made up of $k = 4$), where disabled links have their status set to zero; the capacity of the link (1 Mbps); the current state of the DCN traffic T_{ij}; and finally, flows $f = \left(f.S_r, f.D_s, \lambda_f\right)$ which are described by their source, destination, and speed rate. The output of the optimizer is defined in Equation (10). The output of the solver is the paths defined by the active links L_{ij} for each flow and the current link utilization U_{ij} which increases and decreases according to the traffic input matrix, T_{ij},

$$\begin{pmatrix} \left(L_{ij}\right)_{n\times n}, \left(U_{ij}\right)_{n} \\ \{FR(f,i,j) \mid f \in \mathbb{F}, e_{ij} \in \mathbb{E}\} \end{pmatrix} \leftarrow \mathrm{LinGo}\left(\left(BW_{ij}\right)_{n\times n}, \mathbb{F}, \left(T_{ij}^{\mathrm{input}}\right)_{n\times n}\right). \tag{10}$$

Regular network optimization models determine the optimal (shortest, fastest, cheapest, etc.) route from the source to the destination at a given time. In contrast, our model can determine more paths at a time. In addition, it works like a "multi-layer" network optimization model, where the model finds not only one optimal route for one flow with a source and destination but for multiple input flows at a given time. It defines optimal paths for all the input flows, as well as managing the current state of the network. It gets the current traffic load as an input in parallel with new flow demands, which are then considered when determining the optimal paths.

We inject different burst sizes and numbers of random flows into the DCN in order to evaluate our approach. The results in Table 3 show the global optimal solutions for each status. The number of flows appears in the second column. The size of the model (variables and constraints) increases in parallel with the number of flows. In this state, the model runtime is enormous when there are greater than ≈100 flows active at a time, because of the large number of integer variables. We conclude that the limitation of the IntP model is the long runtime when there are multiple, complex flows of traffic. In response, we develop a heuristic algorithm which has the same goal, i.e., to minimize the number of active links in DCNs, in the next section.

Table 3. LinGo test analysis.

Case	No. of Flows	Objective Value	Total Variables	Integer Variables	Total Constraints	Memory (kB)	Runtime (s)
1	120	35	156,912	155,616	319,249	51,179	9891.79
2	110	32	142,656	141,360	290,341	46,542	275.24
3	100	32	130,992	129,696	266,689	42,747	219.28
4	90	33	118,032	116,736	240,409	38,531	452.84
5	80	30	105,072	103,776	214,129	34,315	45.53
6	70	31	90,816	89,520	185,221	29,678	49.00
7	60	29	77,856	76,560	158,941	25,463	12.38
8	50	29	66,192	64,896	135,289	21,247	16.89
9	40	28	51,936	50,640	106,381	17,031	4.37
10	30	28	38,976	37,680	80,101	12,858	3.22
11	20	24	26,016	24,720	53,821	8599	2.22
12	10	19	14,352	13,056	30,169	4805	1.34
13	5	17	7872	6576	17,029	2697	0.79

5. Proposed Power Consumption Method

As in the proposed IntP Objective Function in Section 4.2, the intuition behind our FPLF algorithm is minimizing the energy consumption of DCN. This is achieved by maximizing the link utilization in Equation (11), which is defined for the aggregation and core layer switches,

$$\max\left(\frac{\sum_{e_{ij}\in\mathbb{E}} T_{ij}}{BW^{th}}\right), \quad \forall e_{ij}. \tag{11}$$

Therefore, the FPLF algorithm forces all switches to use the specified link as long as its bandwidth is underutilized. Otherwise, when the link exceeds the threshold value, 90%, of the bandwidth, the FPLF algorithm redirects flows to the shortest alternative path based on a Dijkstra-like algorithm. Before FPLF starts, comprehensive statistics on traffic within the DCN must be available. For this purpose, we developed a small monitoring model.

5.1. Monitoring Model

As a pre-requirement, to enable FPLF to properly work, we developed a small monitoring model to collect all statistics, i.e, real-time traffic from each node by utilizing the OpenFlow protocol 1.0. The OpenFlow protocol provides extensive statistics like flow, port, and group table statistics. In our case, each link consists of two ports, therefore we collect port statistics by sending the OpenFlow "port stats request messages" to all switches, periodically. When receiving the response messages, the monitoring model extracts the bytes and packets aggregation values. Then, according to the relation equation between those regular intervals, the monitoring model calculates the instantaneous traffic change value for each link,

$$\text{instantaneous traffic} = \text{current state of traffic} - \text{last state of the traffic}. \tag{12}$$

5.2. FPLF-Adaptive Algorithm Components

FPLF incorporates three components. We begin by describing the Link-Utility (LU) calculation component. This component is responsible for calculating the utilization matrix **U**, which stores the utility, U_{ij}, for all the links. We continue by describing the Link-Cost (LC) component, which returns the link-cost matrix **C**. This matrix includes the costs C_{ij} of all the links. Finally, the Fill-Shortest Path (FSP) component forces the forwarding elements

to utilize the specified links as long as the bandwidth BW_{ij} bound is satisfied. The pseudo code of LU component is outlined in Algorithm 1. The input parameters are the traffic demand set, $\mathbb{F}$, and the underlay network graph, $\mathbb{G}$, i.e., a Fat-Tree in our case. When a traffic flow $f = (f.S_r, f.D_s, \lambda_f)$ passes through a link, e_{ij}, its corresponding utility U_{ij} is increased by $\frac{\lambda_f}{BW_{ij}}$. The LU component returns the utilization matrix **U** which describes the links as being overloaded, idle, or under-utilized. The pseudo code of LC is illustrated in Algorithm 2. This algorithm computes the cost value for each link e_{ij}. The input parameters are the utilization matrix **U** and the graph $\mathbb{G}$. According to the utility values U_{ij}, the LC algorithm classifies the links to be either overloaded, idle or under-utilized and it sets a different cost C_{ij} for each link as a result of this classification. The cost values that are generated by this component are used as an input to the FSP component. The pseudo-code for FSP is illustrated in Algorithm 3. The FSP component computes the shortest path for a given flow, $\left(f.S_r, f.D_s, \lambda_f\right)$. The input parameters are the network graph, $\mathbb{G}$, the link-cost matrix, **C**, and the incoming flows $\left(f.S_r, f.D_s, \lambda_f\right)$. The FPLF algorithm fills the preferred links by selecting paths with under-utilization links. As the traffic volume increases, under-utilized links gradually become overloaded links, and the FPLF algorithm compels the FPS component to select idle links to meet the traffic demand. Thus, the FPS component finds the most energy-saving path for the flow, depending on the cost values, namely overload, idle, under-utilization of the links that are obtained from the LC algorithm. It is worth mentioning that if all links become loaded, the FPLF algorithm shares new traffic demand as overload traffic between the links. In terms of algorithm time complexity, the FPLF algorithm has order $O(|\mathbb{S}|^2)$ complexity as it depends on Dijkstra algorithm to pick paths based on the minimum number of hops.

Algorithm 1: Link-Utility.

Input : Fat-Tree topology $\mathbb{G}$, traffic demand $\mathbb{F}$
Output: utilization matrix **U** of all links

U = **0**
foreach *edge* $e_{ij} \in \mathbb{G}$ **do**
  foreach $(f.S_r, f.D_s, \lambda_f) \in \mathbb{F}$ *passing through* e_{ij} **do**
    $U_{ij} \leftarrow U_{ij} + \frac{\lambda_f}{BW_{ij}}$
  end
end
sleep()

Algorithm 2: Link-Cost.

Input : utilization matrix **U**, Fat-Tree topology
Output: link-cost matrix **C**

$\forall\, C_{ij} \leftarrow cost.default()$ ▷ **idle** U_{ij}
foreach *edge* $e_{ij} \in \mathbb{G}$ **do**
  if $U_{ij} > BW^{th}$ **then**
    $C_{ij} \leftarrow cost.high()$ ▷ **overload** e_{ij}
  end
  else
    $C_{ij} \leftarrow cost.low(U_{ij})$ ▷ **underutilization** e_{ij}
  end
end
sleep()

Algorithm 3: Fill-Shortest Path (Dijkstra-like algorithm).

Input : Fat-Tree topology $\mathbb{G}$, flow $\left(f.S_r, f.D_s, , \lambda_f\right)$, link-cost matrix **C**
Output: energy-saving path *ESP*

Set all switches unvisited with weight $+\infty$
Set the weight of Sr to 0
while $\exists$ *unvisited switch* **do**
  $S_i \leftarrow$ unvisited switch with lowest weight W_i
  Add S_i to *ESP*
  Set S_i visited
  if $S_i = d_s$ **then**
    break
  end
  foreach *unvisited* S_j *in neighborhood of* S_i **do**
    $W_j \leftarrow \min(W_j, W_i + C_{ij})$
  end
end

6. Proposed Framework and Implementation

The experimental platform is constructed using the Mininet network emulator [31], and the POX controller which is run using Ubuntu 20.04.2 LTS 64-bit. The network topology is imported from the Fast Network Simulation Setup (FNSS) [32], and the FPLF algorithm is implemented using the POX controller as POX modules. To generate network traffic from servers within a DCN, the Distributed Internet Traffic Generator (D-ITG) [33] platform is used. The architecture is shown in Figure 2.

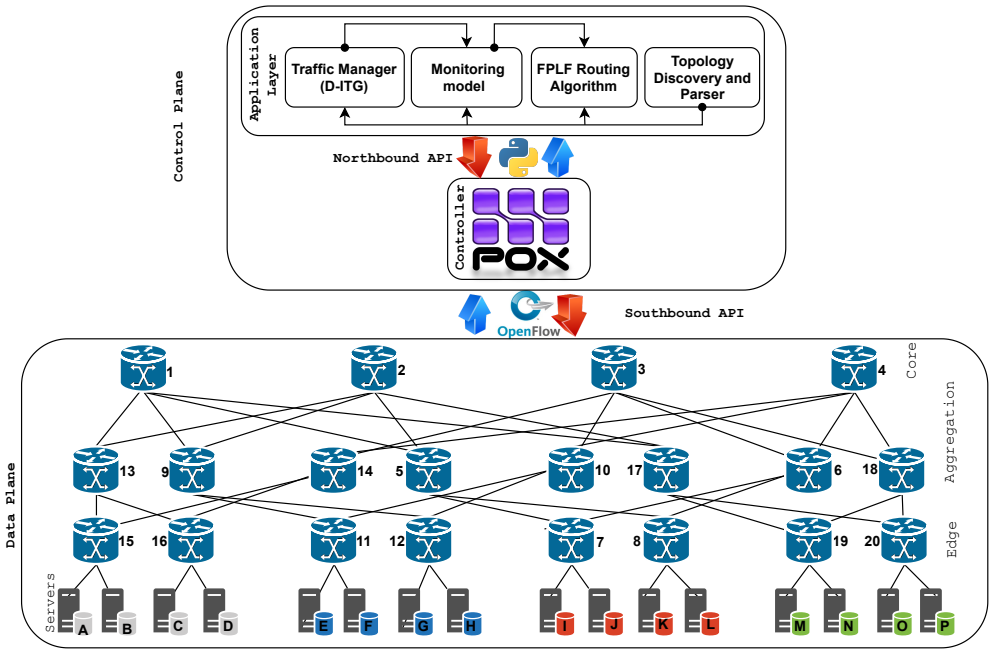


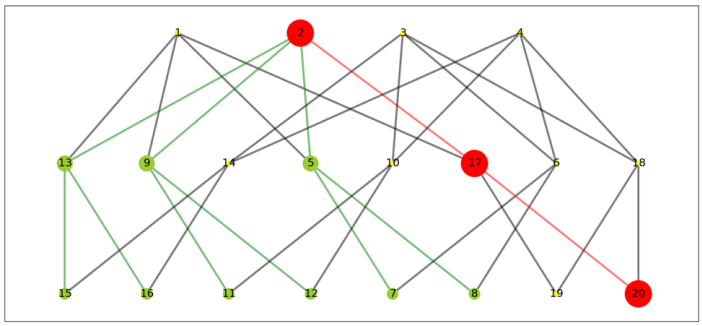
Figure 2. Architecture of the proposed framework and its components: the primary contribution is the FPLF algorithm and the traffic-aware optimizer blocks. OpenFlow was used on the southbound interface and POX Python APIs were used on the northbound interface.

To demonstrate that the FPLF algorithm has an adaptive primary function proportional to traffic demand, we will experiment with different scenarios of traffic demand.

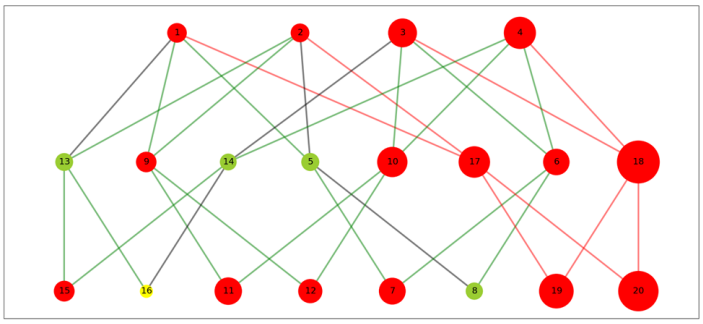
6.1. Experiment Design and Simulation Results

We now explore the possible energy savings of the FPLF algorithm over both low, i.e., multiple small-size flows, and high traffic demand, i.e., multiple large size flows. We considered fat-tree topology with $k = 4$, as depicted in the data plane of Figure 2 in our experiments. We deployed a D-ITG transmitter onto the terminal of each sender server, i.e., A-H, and a D-ITG receiver onto the terminal of the servers M-P. In essence, D-ITG was used to inject high and low traffic load in the simulated network. Two UDP traffic patterns were simulated here, namely, low and high traffic. We set the simulation time to be 2 min for the low traffic case and 8 min for the high traffic. The packet size used for both patterns was 512 bytes. The flow rate of low traffic was constant and set to 10 packets per second, whereas, the high traffic was set to range over 10 to 400 packets per second. Traffic was generated between the servers in Figure 2.

The results show that the FPLF algorithm collected all traffic in one core switch as a best-case when the utility of the links of the paths was below the threshold value, i.e., low traffic demand. This leads to the most significant energy savings because the FPLF algorithm reduces the number of used core switches to the minimum number of links with an idle:active ratio of 1:$\mathbb{E}_{core}$. Figures 3a and 4a show the power saving paths and a time-varying utilization that is lower than the threshold value, respectively.



(**a**) Low traffic load



(**b**) High traffic load.

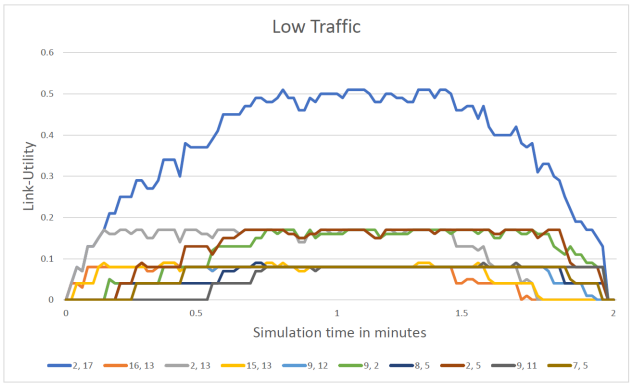
Figure 3. Forward (–), downward (–), and unused (–) paths are illustrated along with the cumulative traffic which is being served by DCN nodes: • > 10 Mb, • 500 kb–10 Mb, • < 500 kb.

Table 4 illustrates the first 33 s from the low traffic scenario events with the responses of the FPLF algorithm for each burst of flows. The remaining 87 s are not shown due to the lack of events, except for the utility of download link (D-L (2,17)) gradually decreasing until the end of the simulation.

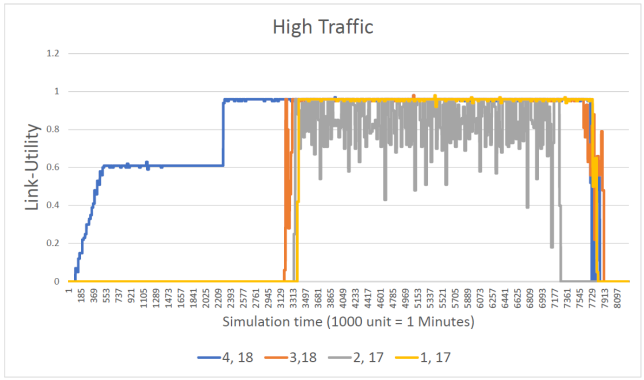
Table 4. Low traffic scenario events that were generated and had server P as the traffic's destination.

Source	Number of Flows	Start Time (s)	End Time (s)	D-L Utility (2,17)	FPLF-Action	Number of the Core Switches
A, B	Single/Multiple small	0	100	increased from 0 to 1.7	Install-ESP	1
C, D	Single/Multiple small	3	105	increased from 1.7 to 0.22	Install-ESP	1
E, F	Single/Multiple small	10	115	increased from 0.22 to 0.25	Install-ESP	1
G, H	Single/Multiple small	15	120	increased from 0.25 to 0.37	Install-ESP	1
I, J	Single/Multiple small	20	120	increased from 0.37 to 0.51	Install-ESP	1
K, L	Single/Multiple small	33	120	fluctuated between 0.51 and 0.47	Install-ESP	1

In the second experimental scenario, we increazed the traffic gradually until the link-utilization value exceeded the threshold. The FPLF algorithm responded to this change by balancing the traffic among core switches to meet the traffic demands as well as to maintain the QoS. This scenario can be considered as the worst-case scenario due to the lower energy saving. In summary, as the demand continued to increase, the FPLF algorithm filled all available links and power savings became minimum because every link in the network was utilized. Figures 3b and 4b show that the load balancing paths and the link-utility of core switches was higher than the threshold value, respectively.



(**a**) Low traffic utility.



(**b**) High traffic utility.

Figure 4. The link utilization is illustrated for both scenarios. The utility is less than the threshold for low traffic and it exceeds the threshold for high traffic scenarios.

Figure 4b clarifies that the FPLP algorithm distributes the traffic according to the number of available core switches. Members of the download links set {(1, 17), (2, 17), (3, 18), (4, 18)} all exceeded the threshold value except for the link, (3, 18), whose link-utilization value fluctuated (illustrated with grey colour). Recall that the IntP model in Section 4.2 required a long time to find the optimal solution for this type of complex set

of flows. In contrast, the FPLF algorithm, which is a heuristic algorithm, found a feasible solution, which explains why the link (3, 18) was not fully utilized. Tables 5 and 6 illustrate the flows patterns, durations and the total events of the high traffic scenario along with the responses of the FPLF algorithm to each burst of flows.

Table 5. High traffic flow details.

Source	Destination	Flows Description
A,B,C	M	Between 0 to 11 s, started with multiple small flows, after 140 s from the time of simulation, burst with large flow, i.e., high traffic
D,E,F	N	Between 0 to 11 s, started with multiple small flows, after 190 s from the time of simulation, burst with large flow, i.e., high traffic
G,H,I	O	Between 9 to 30 s, started with multiple small flows, after 190 s from the time of simulation, burst with large flow, i.e., high traffic
G,K,L	P	Between 21 to 28 s, started with multiple small flows, after 190 s from the time of simulation, burst with large flow, i.e., high traffic

Table 6. High traffic event details.

Events Time (s)	D-L Utility (1, 18)	D-L Utility (3, 18)	D-L Utility (2, 17)	D-L Utility (4, 18)	FPLF-Action	Number of the Core Switches
0 TO 136	0	0	0	0 increased to 0.6	Install-ESP	1
136 TO 190	0	0	0	0.6 increased to 0.96	Install-ESP	1
190 TO 200	0	0.3 increased to 0.96	0	0.96	Install-ESP	2
200 TO 202	0	0.96	0 increased to 0.96	0.96	Install-ESP	3
202 TO 205	0 increasing to 0.96	0.96	0.96	0.96	Install-ESP	4
205 TO 440	0.1.8 fluctuated between 0.96	0.96	0.96	0.96	No action	4
440 TO 466	0	0.96	0.96	0.96	No action	3
466 TO 470	0	0.96 decreased to 0.57	0.96	0.96	No action	3
470 TO 486	0	0.57 decreased to 0	0.96 decreased to 0	0.96 decreased to 0	No action	0

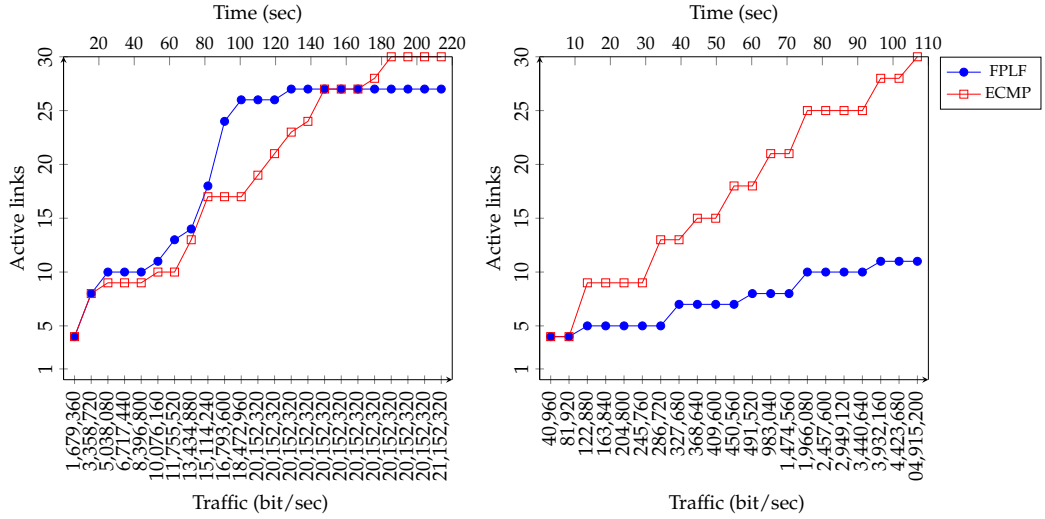
The cumulative traffic values in Figure 3a illustrate how the switches are utilized during the simulation time depending on the utility of the link. In the first scenario, the sub-path nodes $2 \rightarrow 17 \rightarrow 20$ have high cumulative values due to the energy-saving path returned by the FPLF algorithm. Moreover, in the second scenario because the traffic was gradually increased from low to high, Figure 3b shows different cumulative values to core switches ($T2 < T1 < T3 < T4$), corresponding to link utilization values in Figure 4b, where core switch number 4 and the download switches (i.e., switches located along the downward path) have high cumulative values. This happened because the FPLF algorithm selected an energy-saving path in the first 3 min of the simulation, and after that the algorithm started to gradually open new paths to accommodate the traffic and to meet the QoS requirement.

6.2. Performance Evaluation

In order to gauge the performance of the proposed method, we compared our work to the ECMP approach, which is described in [22]. Two metrics were considered: (1) power consumption and (2) number of utilized links.

The FPLF algorithm outperformed ECMP in both cases (low and high traffic). In low traffic cases, which is illustrated in Figure 5b, the FPLF algorithm reduced the number of links by installing energy-saving paths in DCN's switches. Intuitively, the number of utilized links and switches should increase proportionally with increasing traffic to meet QoS targets. That is what happens in Figure 5a, which illustrates the low latency

of the FPLF algorithm with a sudden spiking of traffic in a short time, compared to the ECMP algorithm, to accommodate the increase in traffic. While the FPLF algorithm returns 16 paths with 27 links, ECMP in comparison returns 16 paths with 30 links. This suggests that the FPLF algorithm is more traffic-aware. Since the number of downward links from core switches is four, as shown in Figure 3, bursty high traffic from all other servers leads to these four links being filled, and adding any new upload link is useless; therefore, the algorithm settles on 27 links.



(**a**) High traffic. (**b**) Low traffic.

Figure 5. The DCN link usage under different traffic load scenarios.

In this study, two important QoS criteria were considered: (1) the algorithm response time, i.e., latency and (2) the number of dropped packets. Figure 5a shows FPLF has a high response (i.e., low latency) to unexpected changes in traffic over time. We observed this when the traffic reached the maximum value of 20 Mbps. The FPLF algorithm stabilized with the fixed number of links (27) at simulation time, 130 s, until the end of the experiment. In contrast, ECMP stabilized only after 190 s. That means the ECMP experienced a delay of 70 s before it achieved the same QoS as the FPLF algorithm. However, this delay decreased to 10 s in the low traffic scenario in Figure 5b. One of the crucial metrics that describes the performance of FPLF in the high traffic case is the number of dropped packets. Therefore, during the same power consumption level testing, we used DITG decoder logs to record the results in Tables 7 and 8. Overall, it can be concluded from the above tables that FPLF has a lower percentage of dropped packets compared to ECMP. Minimizing the number of links reduces the amount of power used by turning off the ports and switches it connects. How the percentage of saved links is computed is described in Equations (13) and (14)

$$\text{Active Links } = \sum_{e_{ij} \in \mathbb{E}} L_{ij}, \tag{13}$$

$$\text{Links saved } = \left(1 - \frac{ActiveLinks}{|\mathbb{E}|}\right) \cdot 100\%. \tag{14}$$

According to the above metrics, the FPLF algorithm exhibits the best energy saving in all traffic volume scenarios. The least of the best energy saving is 15.625% in high traffic and 65.625% in low traffic, while the state-of-the-art algorithm recorded a 6.25% energy saving in both scenarios. Moreover, the energy-saving ratio (Equation (15)) between the power consumption of both ECMP and FPLF in terms of active links under the same traffic conditions is 10% in high traffic and 63.3% in low traffic. The energy-saving ratio is defined as

$$\text{energy-saving ratio} = \left[1 - \frac{\text{FPLF Active Links}}{\text{ECMP Active Links}}\right] \cdot 100\%. \quad (15)$$

We draw the following conclusions based on the results reported so far. The FPLF algorithm can find application in the DCNs which serve various traffic patterns (i.e., various traffic loads). For instance, during night times (e.g., when the traffic load is low), the FPLF algorithm has the chance to save up to 65.625% of the total energy consumed, which reduces network operating costs. During periods of large usage, such as rush hour, e.g., when the traffic load is high, the FPLF algorithm can distribute traffic load between switches at low latency, (i.e., the transition time between traffic patterns), which helps to maintain the network performance.

Table 7. Number of dropped packets for the ECMP algorithm.

Destination Server	Total Time (S)	Total Received Packets	Packets Dropped-ECMP	Packets Dropped %
M	429	58,614	128	0.22%
N	326.6	53,329	157	0.29%
O	335.6	53,413	206	0.38%
P	439.7	59,053	415	0.70%

Table 8. Number of dropped packets for the FPLF algorithm.

Destination Server	Total Time (S)	Total Received Packets	Packets Dropped-FPLF	Packets Dropped %
M	414	58,360	152	0.26%
N	322	56,705	117	0.21%
O	330	59,671	115	0.19%
P	423	62,706	114	0.18%

6.3. Limitations

As a result of these experiments, we discuss the disadvantages of the FPLF algorithm. It is a heuristic algorithm and so it is unlikely to achieve the optimal solution. In addition, in its current form, the FPLP algorithm is unable to distribute the traffic load between switches according to the type of traffic that constitutes the flows, i.e., DNS, Ping, voice, video and so on. Figure 6 shows the core switch 4 in the proposed framework as it forwards different flows from the aggregation layer. It is not aware of fault-tolerance values for those flows. Consequently, this might lead to unfair sharing of switches because of the possibility of the FPLF algorithm combining the low or high flows' fault-tolerance on one switch. We believe that combining machine learning classification techniques, for example, based on the features used in approaches such as [34] and the classifiers in [35], with the FPLF algorithm would lead to a more fair sharing of switches. In terms of the FPLP algorithm's time complexity, this might be reduced if the FPLF searched subgraphs for paths instead of the entire graph using a variant of the approach in [36]. The challenge here lies in how to guarantee that we can obtain a path from multiple sub-graphs.

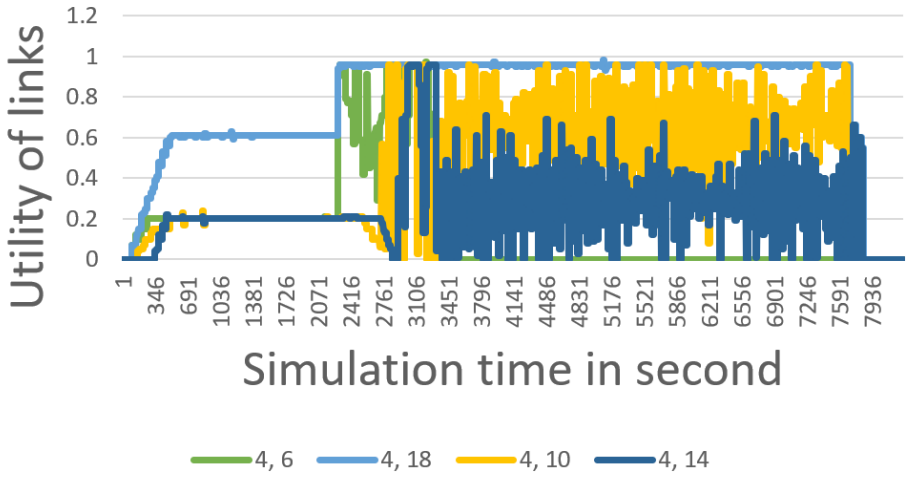


Figure 6. Different fault-tolerance flows sharing one core switch.

7. Conclusions

In this paper, we presented an IntP model for traffic and energy-aware routing based on link utility information for SDN-based DCNs. We tested the IntP model by using the high-level linear programming tool LinGO. Multiple and complex flow cases were used, which showed that the proposed model could determine more paths at a time, although there were some limitations in the cases of high flow volumes. We also proposed a link utility-based heuristic algorithm called FPLF that struck a compromise between energy saving and performance. The FPLF algorithm was evaluated using a real DCN topology with high and low traffic volumes. The experimental evaluation was executed by using the high-quality emulator, Mininet. The FPLF algorithm was implemented as a model using the POX controller. Comparative experimental results showed that FPLF outperformed the existing routing algorithm, ECMP, in balancing how high traffic levels were served across the DCN topology. It achieved up to 10% energy savings in high traffic and 63.3% in low traffic scenarios. As future work, we will enhance the computational performance associated with solving an IntP problem in this setting by considering some predefined estimates of the objective value using heuristic approaches such as those outlined in [37–39]. We are also planning to build a quick definition model for the feasible solution so that we can provide the theoretical upper bound for the objective value. This predefined bound will be used to reduce the number of iterations required to solve the IntP problem with a branch and bound algorithm. Moreover, we aim to develop a new flow estimation model to enhance the FPLF performance as well as to analyze the effect of the model in terms of energy-saving and scalability by utilizing community detection and path anatomy techniques [36].

Author Contributions: Conceptualization, M.N.; Methodology, A.M. and R.d.F.; software of FPLF algorithm, M.N. and A.M.; software of InP model, A.R.; Formal analysis, A.R. and M.N.; Investigation, M.N. and G.K.; writing—original draft preparation, M.N. and A.M.; writing—review and editing, R.d.F. and G.K.; supervision, G.K.; project administration, G.K. and R.d.F.; funding acquisition, R.d.F. All authors have read and agreed to the published version of the manuscript.

Funding: This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/SIRG/3459.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Choumas, K.; Giatsios, D.; Flegkas, P.; Korakis, T. SDN Controller Placement and Switch Assignment for Low Power IoT. *Electronics* **2020**, *9*, 325. [CrossRef]
2. Nsaif, M.R.; Al-Haboobi, A.S.; Rabee, F.; Alasadi, F.A. Reliable Compression Route Protocol for Mobile Crowd Sensing (RCR-MSC). *J. Commun.* **2019**, *14*, 170–178. [CrossRef]
3. Rabee, F.; Al-Haboobi, A.; Nsaif, M.R. Parallel three-way handshaking route in mobile crowd sensing (PT-MCS). *J. Eng. Appl. Sci.* **2019**, *14*, 3200–3209. [CrossRef]
4. Heller, B.; Seetharaman, S.; Mahadevan, P.; Yiakoumis, Y.; Sharma, P.; Banerjee, S.; McKeown, N. Elastictree: Saving energy in data center networks. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010; Volume 10, pp. 249–264.
5. Assefa, B.G.; Özkasap, Ö. A survey of energy efficiency in SDN: Software-based methods and optimization models. *J. Netw. Comput. Appl.* **2019**, *137*, 127–143. [CrossRef]
6. Bertoldi, P. A market transformation programme for improving energy efficiency in data centres. In Proceedings of the ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA, USA, 17–22 August 2014.
7. Avgerinou, M.; Bertoldi, P.; Castellazzi, L. Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency. *Energies* **2017**, *10*, 1470. [CrossRef]
8. Tatchell-Evans, M.; Kapur, N.; Summers, J.; Thompson, H.; Oldham, D. An experimental and theoretical investigation of the extent of bypass air within data centres employing aisle containment, and its impact on power consumption. *Appl. Energy* **2017**, *186*, 457–469. [CrossRef]
9. Staessens, D.; Sharma, S.; Colle, D.; Pickavet, M.; Demeester, P. Software defined networking: Meeting carrier grade requirements. In Proceedings of the 2011 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), Chapel Hill, NC, USA, 13–14 October 2011; pp. 1–6.
10. Thanh, N.H.; Nam, P.N.; Truong, T.H.; Hung, N.T.; Doanh, L.K.; Pries, R. Enabling experiments for energy-efficient data center networks on openflow-based platform. In Proceedings of the 2012 Fourth International Conference on Communications and Electronics (ICCE), Hue Royal City, Vietnam, 1–3 August 2012; pp. 239–244.
11. Riekstin, A.C.; Januário, G.C.; Rodrigues, B.B.; Nascimento, V.T.; Carvalho, T.C.; Meirosu, C. Orchestration of energy efficiency capabilities in networks. *J. Netw. Comput. Appl.* **2016**, *59*, 74–87. [CrossRef]
12. Shirayanagi, H.; Yamada, H.; Kono, K. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. *IEICE Trans. Inf. Syst.* **2013**, *96*, 2055–2064. [CrossRef]
13. Galán-Jiménez, J.; Polverini, M.; Cianfrani, A. Reducing the reconfiguration cost of flow tables in energy-efficient software-defined networks. *Comput. Commun.* **2018**, *128*, 95–105. [CrossRef]
14. Wang, S.H.; Huang, P.P.W.; Wen, C.H.P.; Wang, L.C. EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks. In Proceedings of the The International Conference on Information Networking 2014 (ICOIN2014), Phuket, Thailand, 10–12 February 2014; pp. 220–225.
15. Dalvandi, A.; Gurusamy, M.; Chua, K.C. Power-efficient resource-guaranteed VM placement and routing for time-aware data center applications. *Comput. Netw.* **2015**, *88*, 249–268. [CrossRef]
16. Kang, N.; Liu, Z.; Rexford, J.; Walker, D. Optimizing the "one big switch" abstraction in software-defined networks. In Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, Santa, Barbara, CA, USA, 9–12 December 2013; pp. 13–24.
17. Kanizo, Y.; Hay, D.; Keslassy, I. Palette: Distributing tables in software-defined networks. In Proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 545–549.
18. Rifai, M.; Huin, N.; Caillouet, C.; Giroire, F.; Lopez-Pacheco, D.; Moulierac, J.; Urvoy-Keller, G. Too many SDN rules? Compress them with MINNIE. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–7.
19. Izima, O.; de Fréin, R.; Malik, A. A Survey of Machine Learning Techniques for Video Quality Prediction from Quality of Delivery Metrics. *Electronics* **2021**, *10*, 2851. [CrossRef]
20. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
21. Hopps, C. Analysis of an Equal-Cost Multi-Path Algorithm. *RFC* **2000**, *2992*, 1–8.
22. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic flow scheduling for data center networks. In Proceedings of the 7th USENIX conference on Networked systems design and implementation, San Jose, CA, USA, 28–30 April 2010; Volume 10, pp. 89–92.
23. Luo, J.; Zhang, S.; Yin, L.; Guo, Y. Dynamic flow scheduling for power optimization of data center networks. In Proceedings of the 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), Shanghai, China, 13–16 August 2017; pp. 57–62.
24. Vu, T.H.; Luc, V.C.; Quan, N.T.; Nam, T.M.; Thanh, N.H.; Nam, P.N. The new method to save energy for Openflow Switch based on traffic engineering. In Proceedings of the 2014 2nd International Conference on Electronic Design (ICED), Penang, Malaysia, 19–21 August 2014; pp. 309–314.

25. Assefa, B.G.; Ozkasap, O. Link utility and traffic aware energy saving in software defined networks. In Proceedings of the 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkey, 5–8 June 2017; pp. 1–5.
26. Orlowski, S.; Wessäly, R.; Pióro, M.; Tomaszewski, A. SNDlib 1.0—Survivable network design library. *Netw. Int. J.* **2010**, *55*, 276–286. [CrossRef]
27. Assefa, B.G.; Özkasap, Ö. Resdn: A novel metric and method for energy efficient routing in software defined networks. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 736–749. [CrossRef]
28. Malik, A.; Fréin, R.d.; Ke, C.H.; Alyasiri, H.; Izima, O. Bayesian Adaptive Path Allocation Techniques for Intra-Datacenter Workloads. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; pp. 1–8. [CrossRef]
29. Kaup, F.; Melnikowitsch, S.; Hausheer, D. Measuring and modeling the power consumption of OpenFlow switches. In Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop, Rio de Janeiro, Brazil, 17–21 November 2014; pp. 181–186.
30. Inc, L.S. *LINGO the Modeling Language and Optimizer*; Technical Support; LINDO Systems Inc.: Chicago, IL, USA, 2020. Available online: http://www.lindo.com (accessed on 1 August 2021).
31. Lantz, B.; Heller, B.; McKeown, N. A network in a laptop: Rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, New York, NY, USA, 20–21 October 2010; pp. 1–6.
32. Saino, L.; Cocora, C.; Pavlou, G. A Toolchain for Simplifying Network Simulation Setup. In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, Cannes, France, 5–7 March 2013; SIMUTOOLS '13; ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Brussels, Belgium, 2013.
33. de Donato, M.; Dainotti, A.; Avallone, S.; Pescap, A. *D-ITG 2.8.1 Manual*; COMICS (COMputer for Interaction and CommunicationS). Available online: http://www.traffic.comics.unina.it/software/ITG/ (accessed on 1 August 2021).
34. De Fréin, R. Source Separation Approach to Video Quality Prediction in Computer Networks. *IEEE Commun. Lett.* **2016**, *20*, 1333–1336. [CrossRef]
35. de Fréin, R.; Izima, O.; Malik, A. Detecting Network State in the Presence of Varying Levels of Congestion. In Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, Gold Coast, Australia, 25–28 October 2021; pp. 1–6.
36. Malik, A.; de Fréin, R.; Aziz, B. Rapid restoration techniques for software-defined networks. *Appl. Sci.* **2020**, *10*, 3411. [CrossRef]
37. Savelsbergh, M.W.; Sigismondi, G.C.; Nemhauser, G.L. *Functional Description of MINTO, a Mixed Integer Optimizer*; Eindhoven University of Technology: Eindhoven, The Netherlands, 1991.
38. Woodruff, D.L. A chunking based selection strategy for integrating meta-heuristics with branch and bound. In *Meta-Heuristics*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 499–511.
39. Septién, J.; Mozos, D.; Tirado, F.; Hermida, R.; Fernández, M. Heuristics for branch-and-bound global allocation. In *EURO-DAC*; IEEE Computer Society Press: Washington, DC, USA, 1992; pp. 334–340.