electronics

MDPI

Article

QoS-Aware Power-Optimized Path Selection for Data Center Networks (Q-PoPS)

Mohammed Nsaif [1,2,3,*], Gergely Kovásznai [4], Ali Malik [5] and Ruairí de Fréin [5]

[1] Department of Information Technology, University of Debrecen, 4032 Debrecen, Hungary
[2] Department of Business Management, University of Kufa, Al-Najaf 540011, Iraq
[3] Department of Computer Science, Al-imam University College, Balad 34011, Iraq
[4] Department of Computational Science, Eszterhazy Karoly Catholic University, 3300 Eger, Hungary; kovasznai.gergely@uni-eszterhazy.hu
[5] School of Electrical and Electronic Engineering, Technological University Dublin, D07 EWV4 Dublin, Ireland; ali.malik@tudublin.ie (A.M.); ruairi.defrein@tudublin.ie (R.d.F.)
* Correspondence: mohammed.nsaif@mailbox.unideb.hu

Abstract: Data centers consume significant amounts of energy, contributing indirectly to environmental pollution through greenhouse gas emissions during electricity generation. According to the Natural Resources Defense Council, information and communication technologies and networks account for roughly 10% of global energy consumption. Reducing power consumption in Data Center Networks (DCNs) is crucial, especially given that many data center components operate at full capacity even under low traffic conditions, resulting in high costs for both service providers and consumers. Current solutions often prioritize power optimization without considering Quality of Service (QoS). Services such as video streaming and Voice over IP (VoIP) are particularly sensitive to loss or delay and require QoS to be maintained below certain thresholds. This paper introduces a novel framework called QoS-Aware Power-Optimized Path Selection (Q-PoPS) for software-defined DCNs. The objective of Q-PoPS is to minimize DCN power consumption while ensuring that an acceptable QoS is provided, meeting the requirements of DCN services. This paper describes the implementation of a prototype for the Q-PoPS framework that leverages the POX Software-Defined Networking (SDN) controller. The performance of the prototype is evaluated using the Mininet emulator. Our findings demonstrate the performance of the proposed Q-PoPS algorithm in three scenarios. Best-case: Enhancing real-time traffic protocol quality without increasing power consumption. midrange-case: Replacing bottleneck links while preserving real-time traffic quality. Worst-case: Identifying new paths that may increase power consumption but maintain real-time traffic quality. This paper underscores the need for a holistic approach to DCN management, optimizing both power consumption and QoS for critical real-time applications. We present the Q-PoPS framework as evidence that such an approach is achievable.

Keywords: data center networks; energy efficiency; quality of service; software-defined networking; real-time traffic; power optimization; QoS-aware path selection

Citation: Nsaif, M.; Kovásznai, G.; Malik, A.; de Fréin, R. QoS-Aware Power-Optimized Path Selection for Data Center Networks (Q-PoPS). *Electronics* **2024**, *13*, 2976. https://doi.org/10.3390/electronics13152976

Academic Editor: Martin Reisslein

Received: 10 June 2024
Revised: 15 July 2024
Accepted: 23 July 2024
Published: 28 July 2024

1. Introduction

Demand for digital services is growing rapidly. Since 2010, the number of Internet users worldwide has doubled, and global Internet traffic has expanded 20-fold. However, the proliferation of data centers and transmission networks that support digitalization has resulted in increased energy consumption [1]. According to the authors of [2,3], this has two significant impacts: (1) higher costs for both service providers and end-users, and (2) environmental concerns related to CO_2 emissions.

To address the challenges of increasing energy demand and emissions, it is crucial to prioritize improving energy usage efficiency. This is particularly important in limiting the growth in energy demand from Data Center Networks (DCNs) and data transmission

networks. Estimates indicate that in 2005, DCN consumed 153 TWh, and by 2010, this consumption had increased to 273 TWh of global electricity use, accounting for between 1 and 1.5% of global electricity demand [4]. Furthermore, according to this research note [5] from the Borderstep Institute, it is projected that the energy consumption of DCNs worldwide may increase by a factor of 40 between 2019 and 2030. Researchers, in collaboration with government and industry have placed considerable effort into improving energy efficiency and procuring sources of renewable energy, to mitigate and to curb energy demands, and emissions growth over the next decade.

In recent years, several studies have focused on managing and reducing energy consumption in DCNs. Various techniques have been proposed to reduce energy consumption in DCNs. Many of these techniques primarily targeted servers, utilizing Dynamic Voltage and Frequency Scaling (DVFS) techniques to manage power, implementing intelligent cooling system management within the computational environment, and employing server consolidation strategies to minimize the number of physical machines used in DCNs. However, fewer studies have focused on the routing layer (network layer). This may be attributed to the limitations imposed by the physical substrates of legacy networks, for instance the inherent limitations of the TCP/IP protocol stack. These limitations are summarized in [6]: (1) limited support for new ideas and innovations; (2) lack of management flexibility, due to the tight coupling of the control plane and data plane, which necessitates changes to both when altering the control plane; (3) a large number of multicast messages are required to construct the routing table of the switches; (4) an increased energy footprint.

Furthermore, sophisticated routing-aware techniques rely the availability of prior knowledge of the network's state. In legacy networks, protocols such as the Simple Network Management Protocol (SNMP) and tools such as IPFIX [7] and NetFlow [8], are used for analyzing the network's status. However, these legacy networks often involve diverse forwarding equipment from different suppliers, resulting in variations in capabilities and configurations. As a result, collecting accurate statistics and monitoring data can become challenging. For instance, the SNMP protocol may encounter discrepancies in the set of supported SNMP agents across different network devices, leading to inconsistencies in the collected statistics. Similarly, in the case of the reporting system, diverse network equipment can hinder the consistent collection of monitoring data.

Software-Defined Networking (SDN) has emerged as a solution for these issues. SDN is a new networking paradigm in which the control plane is physically separated from the data plane. This separation enables network administrators to rapidly design, manage, protect, and optimise network resources using dynamic and automated SDN programs. These programs can be implemented by the network administrators/programmers themselves, because the SDN equipment is programmable and not reliant on proprietary software.

The landscape of affordances provided by SDN has seen the adoption of SDN as an appealing solution for many networking issues. For instance, in terms of efficient energy usage, the literature presents two different energy-saving strategies in the network layer: (1) flow aggregation (FA); and, (2) flow scheduling (FS) techniques [9,10]. These techniques rely heavily on the real-time ability of monitoring applications to perform their roles effectively.

SDN is a centralized networking architecture and it is important to consider that real-time monitoring schemes can introduce overhead on the SDN controller [11]. Monitoring the network in real-time requires the controller to collect, process, and extract data patterns from various network instruments, such as ports, switches, and servers on an ongoing basis. This can lead to increased processing, energy, and memory requirements for the controller, which affects its performance.

This paper proposes a new framework that balances efficient power usage with an acceptable level of Quality of Service (QoS). We focus on real-time traffic delay as the primary QoS metric because Software-Defined Data Center Networks (SD-DCNs) must address the growing volume of delay-sensitive traffic in DCNs, as highlighted in the Cisco Global Cloud Index (2016–2021) [12]. Therefore, the QoS-Aware Power-Optimized Path Selection

(Q-PoPS) framework introduced in this paper aims to effectively manage and optimize the handling of delay-sensitive traffic. This optimization is achieved under the constraint of minimizing DCN energy usage, as described in [10]. We accomplish this by maximizing port utilization and packing multiple flows into a single link whenever possible.

The following is a summary of the paper's contributions:

1. **Integer Linear Programming (ILP) Model**: The paper proposes a new ILP model aimed at optimizing power usage while considering multiple constraints, such as prioritizing real-time traffic applications.
2. **Q-PoPS Framework**: The paper introduces a new framework named Q-PoPS. Q-PoPS leverages the SDN paradigm at the control layer for power-optimized path selection while ensuring QoS targets for critical real-time traffic. A crucial aspect of this framework is a lightweight traffic monitoring model, which focuses only on delay-sensitive traffic in DCN, supported by a traffic classifier.
3. **Real-time Implementation**: The paper presents a series of simulation scenarios in the Mininet emulator to evaluate the framework, demonstrating that it balances efficient power usage with an acceptable level of QoS.

This paper is organized as follows: In Section 2, we delve into the literature, discussing the challenges and solutions in the state of the art for both monitoring and energy optimization. In Section 3, we express the problem statement, highlighting the recent surge in real-time traffic applications in DCNs and how energy strategies impact their performance. Section 4 presents the Integer Linear Programming (InLP) model formulation. We develop the Q-PoPS framework in Section 5. We showcase our experiments and results in Section 6. Lastly, in Sections 7 and 8, we engage in discussions, outlining future directions, and draw conclusions, respectively.

2. Related Work

In this section, we discuss the state-of-the-art power optimization and traffic monitoring strategies that aim to improve energy usage in DCNs.

2.1. Power Optimization

The literature has identified two different energy-saving strategies in the network layer. Firstly, FA techniques involve grouping multiple flows and routing them over the same links. However, these techniques may have an adverse effect on QoS metrics, such as network latency and throughput [10,13–15].

A traffic-proportional energy-efficient framework for SDN, along with a heuristic algorithm that maintains the tradeoff between energy saving and average path length, was proposed in [16]. The paper also presented an InLP formulation for the traffic proportional energy efficiency problem. Experiments were conducted on the Mininet emulator and POX controller, using real-world topologies and traffic traces from Abilene, Atlanta, and Nobel-Germany networks. The results demonstrated that the approach saved up to 50% energy.

The analysis in [13] examined the effect of using different over-subscription factors and Point of Delivery (PoD) sizes on energy consumption. The authors employed several strategies of flow grouping, for instance, first fit, worst fit, and best fit in DCNs.

The proposed strategies were evaluated using different link demand over-subscription ratios (1:1, 1:5, and 1:20), in a fat-tree topology with different network workloads (20%, 50%, and 80%). Results showed savings of up to 70% were possible when the over-subscription factor was 1:20 and the topology size was $k = 12$. In this setting, the variable k refers to the number of logical child links (ports) that a switch has in a specific layer of the network. The value k determines the fan-out of a switch within the topology. However, the paper does not provide enough detail about the technical and simulation process for these results to be independently verified.

The authors in [17] presented a Minimum Energy Consumption (MEC) heuristic algorithm based on three strategies: Random-order Demand (RD-MEC), Biggest Demand First (BDF-MEC), and Smallest Demand First (SDF-MEC). This algorithm aims to ensure

the QoS satisfaction ratio during the energy optimization process. The results demonstrated that the proposed method successfully improves the QoS satisfaction ratio while reducing power consumption. However, the authors did not provide sufficient details about the experimental setup and simulation, and the experiment was conducted outside of an SDN environment.

The idea in [18] contributed to the field of green networking by addressing the challenge of sustainability in network routing. They proposed and analyzed the effectiveness of three different metrics: energy consumption, carbon emission factor (CEF), and non-renewable energy usage percentage (NRE), aiming to reduce energy consumption, carbon emissions, and the use of non-renewable energy sources. Their method uses a genetic algorithm to optimize the routing and bandwidth distribution of the network. The pollution-aware routing (PAR) algorithm they developed reduced CO_2 emissions by 36% compared to shortest path first routing and by 20% compared to energy-based routing.

The study in [19] proposed an algorithm named TrimTree to calculate a minimum active subset that is sufficient for carrying out communication while turning off the remaining network devices. The algorithm also adapts the link data rates of those active subsets in different layers of DCN topologies to the workload. The authors also consider QoS by creating a backup network subset, putting them in sleep mode, and awakening them only upon the arrival of traffic. These subsets switch back to sleep mode in case of low utilization for a specified duration. The results reported that the proposed algorithm saves 50–70% more energy compared to the conventional fat-tree.

To design a new framework with the aim of reducing power consumption in SD-DCNs, the authors in [10] proposed a novel InLP model and a heuristic link utility-based algorithm named FPLF. This stuck a balance between energy consumption and network performance. The InLP model was solved using an optimization tool called LinGo http://www.lindo.com (accessed on 14 September 2023), which can handle both convex and non-convex optimization problems. However, the results revealed that finding the optimal solution was time-consuming. As a result, the authors developed a heuristic algorithm and evaluated its performance using an experimental platform that included the POX controller and the Mininet network emulator. By comparing the proposed method with a state-of-the-art approach, the Equal Cost Multi-Path (ECMP) algorithm, the authors obtained results that demonstrated that the new framework reduced power consumption by up to 10% during high-traffic loads and achieved a 63.3% reduction in power consumption during periods of low traffic load. This indicates the effectiveness of the proposed framework in improving energy efficiency in SD-DCNs. However, the framework has not been tested with real-time protocol application flows, and its effect on their performance when optimizing power remains unexplored.

For non-real-time experimental scenarios, a Deep Reinforcement Learning (DRL)-based energy-efficient routing algorithm called Deep Q-Network Energy-Efficient Routing (DQN-EER) was proposed in [20]. The algorithm learned to select the most Energy-Efficient Paths (EEP) for traffic flows in DCNs. The training of DQN-EER was carried out using a Markov Decision Process (MDP), and Python was used to program DQN-EER. The results indicated that DQN-EER consumes a similar amount of energy as the baseline method, the CPLEX https://github.com/aimms/documentation/blob/master/platform/solvers/cplex.rst#cplex (accessed on 22 January 2024) solver. However, its calculation time is significantly smaller than CPLEX's calculation time, especially in scenarios where the number of flows is large. It is important to note that this technique might not be suitable for real-time simulations due to its time cost.

In [21], a power consumption management scheme for multiple controllers in DCNs was presented that aimed to support Internet of Things (IoT) applications within an SDN framework. The proposed scheme divided the control plane into two layers: the bottom layer, which had multiple controllers, and the top layer, which had a single controller. The scheme was comprised of two components, namely the Strategy For Nash Equilibrium Point (SFNEP) component and the Strategy For Energy Optimal (SFEO) component. The

SFNEP component was designed to achieve the Nash Equilibrium Point in the bottom controllers and the SFEO component was responsible for guiding the sleep controllers to achieve the energy efficiency target by using the SFNEP component. The bottom controllers were considered to be non-cooperative players, which led to a decrease in the calculation load of the top controller. Simulation results demonstrated that the proposed scheme obtained more sleep controllers, improving energy efficiency in the DCNs.

The papers discussed so far describe solutions for optimizing the power usage in DCNs. In the context of the trends forecasted by Cisco, it is possible that flow consolidation could lead to increased path latency or packet loss, which needs to be tackled.

FS techniques operate within a free collision domain by scheduling flow transmission within the DCN. These approaches have been explored as a potential solution to improve energy efficiency. FS allows flows to monopolize all the links along their path, enabling them to utilize their full capacity. This approach may suit bandwidth-intensive applications, such as [9,22,23].

The authors in [24] claimed that flow consolidation methods might not be effective for network-limited flows, whose throughputs are elastic and depend on the competing flows. To address this issue, they proposed an FS approach called Willow, which considers both the number of switches involved and their active working durations. Willow is a greedy approximate algorithm that is designed to behave efficiently when it is used online. The results demonstrated that Willow can achieve network energy savings of up to 30% when it is compared with ECMP scheduling in typical settings. However, the paper did not consider the impact of Willow on upper-layer applications.

The study in [25] considered the completion time as a constraint for scheduling flows. They suggested an algorithm named Bandwidth-aware Energy Efficient Routing (BEERs) that worked by first grouping flows with the same deadline together as harmonic flows. The algorithm scheduled these flows simultaneously to minimize the energy consumption of the DCNs. A limitation of the work is that the authors did not explicitly point out how the algorithm determined the size of the flows. In an extension of this work in [26], the authors conducted extensive experiments with two types of topology. The results showed that the extended BEERs algorithm reduced overall energy consumption with respect to traffic volume and that it also reduced the average flow completion time.

A dynamic adaptive scheduling algorithm based on power-aware routing and flow preemption was presented in [22]. The algorithm is named Flow Preemption (FLOWP). It achieved energy savings by combining the most EEP selection and setting a tolerance threshold for each flow. The results demonstrated that the FLOWP algorithm successfully reduced energy consumption by up to 30% compared to a baseline FS algorithm.

In this study, our aim is to leverage the advantages of the FA technique. We extend the same consolidation algorithm presented in [10] by combining it with a delay-monitoring model to measure the delay of each flow based on the application layer. This will allow us to find alternative paths for flow delay-sensitive traffic in SD-DCNs during congestion events.

2.2. Monitoring Strategies

Monitoring the network enables administrators to support network activity effectively and to handle the issues that may arise quickly. To better understand network traffic and to enhance the QoS and delivery for different classes of service, a traffic monitoring approach may perform various tasks such as estimating the traffic matrix, identifying elephant flows, and measuring link utilization [27,28]. Compared to legacy network monitoring systems, SDN-focused monitoring solutions provide greater flexibility and monitoring solutions [29] with increased adaptability. Two classes of monitoring strategies may be used to perform monitoring in SDNs, namely *active* and *passive* [30]. Active monitoring, which is typically based on a polling approach, means that the central SDN controller is responsible for sending message requests regularly to a number of targeted forwarding elements on its domain. A number of studies have utilized and examined the performance of active monitoring approaches. For instance, the authors in [31] developed OpenTM,

which sends monitoring messages periodically to estimate the traffic matrix in a network. In passive monitoring (also called switch pushing), network switches automatically send notifications to a central controller whenever a specific data flow is terminated. One of the early examples of research into passive monitoring techniques is Flowsense [32], in which the OpenFlow `packet_in` and `flow_removed` messages, which are automatically generated based on the arrival and removal of network flows, are used as inputs to the monitoring function. Both active and passive monitoring techniques have limitations. On one hand, active monitoring approaches are typically associated with a significant increase in controller resource usage, which results due to periodic querying of network forwarding elements. One benefit of this is that the network manager can obtain a precise estimate of the current network state. On the other hand, the main drawback of passive monitoring approaches is the inability to offer monitoring reports when the flows are in operation. Consequently, the obtained monitoring reports may not necessarily represent the actual state of the network.

In the context of improving energy efficiency, the power optimization models discussed tend to rely on the active monitoring approach to achieve an acceptable trade-off between reduction in power consumption and the resource usage of the SDN controller. In this paper, we adopt an approach which is based on both passive and active methods. The active monitoring approach is used to build a live view of the current state of the network. The passive monitoring approach is used to update the lists of active flows, paths, etc. This combined approach allows Q-PoPS to reduce the DCN power consumption and to main acceptable levels of QoS for critical services.

3. Problem Statement

This section explores the factors driving the increase in real-time traffic within DCNs. It discusses recent advancements in power optimization algorithms along with their limitations, emphasizing the unique challenges presented by fluctuating flow rates during traffic consolidation aimed at optimizing power usage in DCNs.

3.1. Real-Time Traffic Trends in DCNs

We have established that FA-based techniques in the literature may adversely affect QoS metrics, such as latency and throughput [2]. The rate of increase in real-time traffic in DCNs, especially video traffic, reported in a press release published by Cisco in February 2018 is significant. According to the release, video traffic is projected to account for 85% of traffic from DCNs to end-users by 2021, compared to 78% in 2016. Moreover, video streaming is expected to represent 10% of traffic within DCNs by 2021, up from 9% in 2016. These statistics clearly highlight the growing demand for handling video traffic in DCNs [12].

Figure 1 shows information reported by Statista https://www.statista.com/statistics/871513/worldwide-data-created/ (accessed on 3 April 2024), in 2020, which suggested that data generation and replication reached an unprecedented peak, reaching 64.2 zettabytes, surpassing earlier predictions. This surge was mainly driven by increased demands due to the Coronavirus Disease 2019 (COVID-19) pandemic, as more people were working and studying from home frequently. Most of the activities carried out during this period involved real-time stream applications. In the five years leading up to 2025, global data production is projected to exceed 180 zettabytes. There have been some attempts by the Machine Learning (ML) community to address this problem using sequential learning methods for the Quality of Delivery (QoD) classification [33], the deployment of state-of-the-art quantum classifiers [34], and finally, online ML traffic classifiers [35]. However, the performance of these ML algorithms is determined by the resolution achieved by the monitoring solution that delivers the network statistics to the learning algorithm.

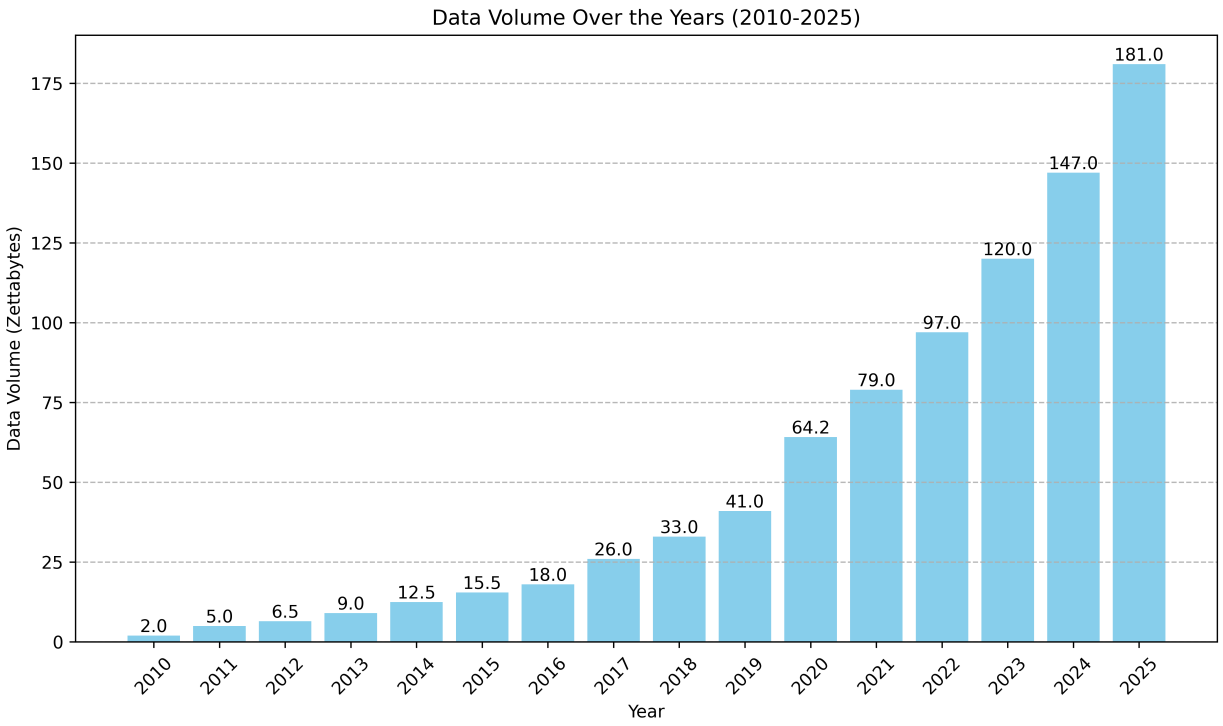


Figure 1. Data volume and the rise in DCN energy consumption: The projected exponential growth in global data volume from 2010 to 2025, measured in zettabytes, is illustrated. This surge in data processing significantly loads DCNs, which are reported to consume roughly 10% of global energy according to the Natural Resources Defense Council.

The trend in Figure 1 underscores the need to develop new monitoring protocols. These new protocols are vital to ensure that DCNs can cope effectively with the increasing demands and challenges posed by the surge in real-time video traffic. To address this issue, developing power management techniques that minimize energy consumption while preserving high-quality performance is of utmost importance. We have contributed to this effort by developing techniques such as Fill-Preferred Link First (FPLF) and Smart-FPLF [1,10,36,37]. The first technique employs lightweight adaptive algorithms to optimize power usage in the SD-DCNs environment. The second technique leverages Deep Learning algorithms to effectively reduce overhead on the SDN controller in addition to managing power usage. The goal of this paper is to enhance power efficiency without compromising the quality of Real-Time Protocol Applications (RTPA) in SD-DCN environments.

3.2. Power Management Techniques in SD-DCNs

Drawing on the existing literature in Section 2, several studies have proposed solutions for optimizing power consumption in DCNs. Nevertheless, it remains a valid concern that implementing power optimization techniques, especially when employing FA techniques, could potentially result in an elevated path delay.

Figure 2 depicts the Fill Preferred Path First (FPLF) algorithm, as presented in [10]. It is comprised of three components: the Link-Utility (LU) component, the Link-Cost (LC) component, and the Fill Shortest Path (FSP) component. In FPLP, the monitoring model gathers statistical information from the DCN to calculate the Traffic Matrix (TM) of the network topology. Then, the TM and the network topology, G, serve as inputs for the LU component, which calculates the link Utilization Matrix (UM). The resulting UM, along with the network topology G, is passed to the LC component to calculate the Cost Matrix (CM). In the final step, the CM together with the network topology, G, and the traffic demand, $\mathbf{F}$, as passed as inputs to the FSP component to calculate the Energy-Efficient Shortest Path (P).

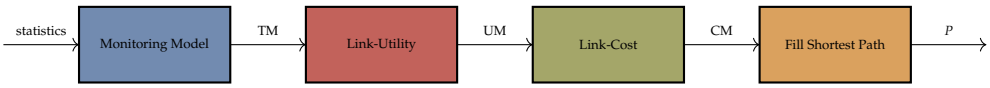


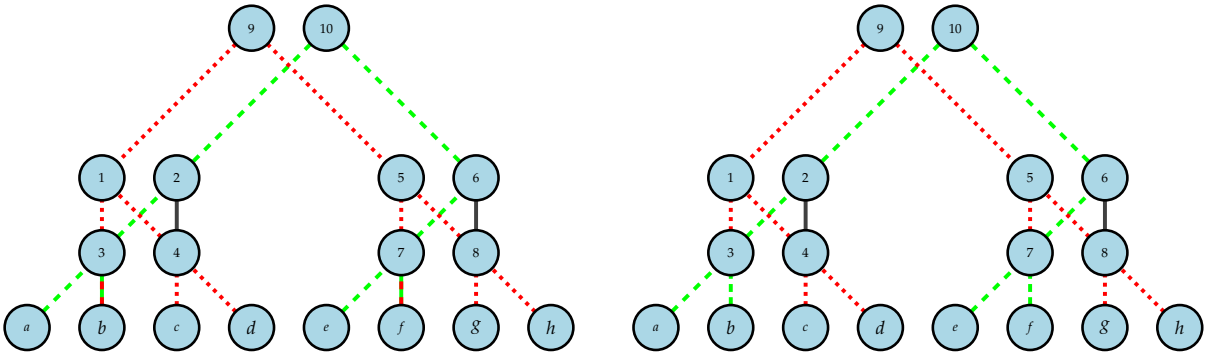
Figure 2. Flow of control of FPLF: FPLF operates in a sequential manner. The output from each of three stages feeds into the next.

We focus on the performance of FPLF as a case study. Our objective is to demonstrate the tangible impact of such an algorithm on the performance of the RTPA. Therefore, we extend the FPLP algorithm, which was introduced in [10], to find alternative paths that meet the RTPA criterion. This is accomplished by introducing a delay measurement algorithm for RTPA flows in DCNs based on the 3-tuple, i.e., link, transport, and application layers.

3.3. Variability in Flow Rates and Path Adaptation

Variability in flow rates, λ, and their associated probability distributions can result in a simultaneous increase in the speeds of multiple flows, driving them toward their peak values. Additionally, optimization algorithms may aggregate flows that exceed a certain threshold due to the difficulty of measuring the bandwidth demand in advance for each flow, leading to heightened delays. This effect is especially pronounced when dealing with time-sensitive traffic. Consequently, it is necessary to identify alternative paths that can accommodate these flows. This could involve actions such as activating new switches and ports, potentially increasing power consumption (worst-case scenario), or utilizing existing active paths that uphold acceptable delays without increasing power consumption (best-case scenario).

To demonstrate our approach, we present an example of the best-case scenario in Figure 3. In this scenario, we constructed a fat-tree topology consisting of 10 switches and 8 servers. All links were assumed to have a uniform bandwidth, b, with the speed 1 Mbps. The scenario was evaluated for 60 s. We assumed that at the beginning of the evaluation period, e.g., at the $t_1 = 1$ s, three flows were introduced. Two flows went between nodes a, and e, e.g., $f(a, e)$, and one flow went between b, and f, e.g., $f(b, f)$. These flows had completion times of 20 s, 50 s, and 50 s, respectively.



(**a**) Pure consolidation method (**b**) RTP-aware consolidation method

Figure 3. The pure consolidation method and the RTP-aware consolidation method of power optimization subject to an RTP-flow delay constraint are illustrated. In sub-figure (**a**): [▬]: Represents the consolidation of three flows. [▬]: Represents the consolidation of 4 flows on segment path $(1, 9, 5)$ from different sources. [▬]: Signifies the sum of 2 flows. In sub-figure (**b**): [▬]: Represents the consolidation of (two plus the RTP flow). [▬]: Represents the consolidation of 3 flows because the RTP flow was moved to an alternative path, represented by [▬].

The FPLF algorithm selected paths that share the same core nodes, such as $(3, 2, 10, 6, 7)$, and combined the flows onto a single path. These flows were sufficiently bandwidth-intensive to saturate the path and the result was a high utilization rate of 100%. At time $t = 10$ s, we introduced additional flows, including an RTP flow class, which is denoted as $f(b, f)$, as shown in Figure 3a. The FPLF algorithm opened a new path in red,

$(b, 3, 1, 9, 5, 7, f)$, to accommodate the new request and marked its corresponding links as under-utilized. This decision was taken due to the over-utilization of the initial path, $(3, 2, 10, 6, 7)$. At time $t = 30$ s, we introduced the flows $f(c, g)$, $f(d, g)$, and $f(d, h)$. FPLF took the action of merging them with the RTP flow on the links $(1, 9, 5)$ and returned the path $(4, 1, 9, 5, 8)$, in order to optimize power usage.

Consider the following analysis. If any delays are encountered on the path $(b, 3, 1, 9, 5, 7, f)$, rerouting traffic onto an existing path that meets requirements becomes a viable option. A candidate path is $(3, 2, 10, 6, 7)$, because one of the flow $f(a, e)$ ended within the first 20 s of this scenario. This action rendered the path $(3, 2, 10, 6, 7)$ under-utilized, as illustrated in Figure 3b. In this scenario, the flow, $f(b, f)$, would be rerouted along the path which is illustrated in green, e.g., $(3, 2, 10, 6, 7)$, due to its RTP flow classification.

This motivational example provides the encouragement to investigate all scenarios of power consumption that a DCN needs to consider to ensure that RTPA meets its associated QoS criteria. A consequence of this is making decisions such as turning on new ports/switches or efficiently utilizing the currently active subset. It is worth noting that in our current study, we assume that all OpenFlow switches can transition into sleep mode and turn on based on traffic load states. We leverage technologies like Sleep-on-Idle (SoI) to conserve power when switches are not actively in use and Wake-on-Arrival (WoA) to activate them promptly when they are needed, e.g., to forward flows. Notably, we do not consider the effects of the transition time in this context [24].

4. Problem Formulation and Modeling

The level of power consumption of SD-DCN routing mechanisms is varied. It is typically based on how the switches involved in forwarding flows operate. These methods can be classified using two main categories: (1) dynamic, which assesses the power used by active links (ports), and (2) static, which calculates the total power consumed by various components like chassis, fans, and switching fabric [36]. This study focuses on dynamic approaches.

Forward and backward traffic flows are delivered on a directed weighted graph, $G(S, E)$. The vertex set is denoted by $S = \{s_1, s_2, \ldots, s_n\}$, and the edge set is denoted by $E \subseteq \{e_{ij} \mid s_i, s_j \in S\}$. The ith OpenFlow switch is denoted as s_i. In the graph, G, the edge, e_{ij}, represents a link that connects the ith and jth switches.

The state of the network links, L_{ij}, are modeled to be either `ON`, e.g., $L_{ij} = 1$, when a link is active, or `OFF`, when a link is inactive, e.g., $L_{ij} = 0$. We define the link to be active when at least one flow is passing over it. The notation that is used in the current paper is tabulated in Table 1.

The Q-PoPS optimization problem extends the optimization problems described in [10,36]. It incorporates additional constraints to ensure that Q-PoPS meets the delay requirements of RTP flows. The Q-PoPS objective function seeks to minimize the sum of the active links, where L_{ij} indicates if a link is active or inactive,

$$\min : \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij}. \tag{1}$$

The traffic volume over the edge, e_{ij}, is denoted as T_{ij}. The traffic volume is a positive-valued variable. We introduce a side constraint that ensures that a link in the solution can be activated if the traffic that traverses it is non-zero. The traffic is expressed here as a fraction of the bandwidth of the link, b_{ij}, where the ratio is $\frac{T_{ij}}{b_{ij}}$ and the resulting side-constraint is

$$\frac{T_{ij}}{b_{ij}} \leq L_{ij}, \quad \forall e_{ij} \in E. \tag{2}$$

If the flow, f, which is a member of the set of all flows, $\mathbf{F}$, passes over the link e_{ij}, then $FR(f,i,j) = 1$; otherwise, it is 0. The next link constraint ensures that flows should only pass over active links. This is expressed as the inequality

$$FR(f,i,j) \leq L_{ij}, \quad \forall f \in \mathbf{F}, \forall e_{ij} \in E, \tag{3}$$

which states that the Boolean, $FR(f,i,j)$, should be less than or equal to the value assigned to L_{ij}.

Table 1. Conventional names of parameters used in problem formulation.

Parameters	**Definitions**
D_{ij}	Delay on link e_{ij} in ms.
D_f	Flow delay, where the set of flow delays is $\mathbf{D}$, and $D_f \in \mathbf{D}$.
H_f	Identifies the flow based on the 3-tuple (transport layer, network layer, link layer), where $H_f \in \mathbb{H}$, where the set of all flow ids is $\mathbb{H}$.
$\mathbf{F}$	Is the set of all flows. A flow consists of ths terms, $f = \left(f.S_r, f.D_s, \lambda_f, H_f, D_f\right) \in \mathbf{F}$. The source is denoted $f.S_r \in S$. The flow destination is denoted, $f.D_s \in \mathbb{S}$. Its bit rate is denoted λ_f. The flow ID is H_f, and the flow delay is D_f.
$\mathbf{P}$	The set of energy-efficient shortest paths is denoted $\mathbf{P}$, where a path, $P = \{s'_1, \ldots, s'_k\} \in \mathbf{P}$ is represented by a set of switches between the servers of DCNs, where $\forall s'_i \in S$.
$\mathbf{F_r}$	The set of RTP flows is denoted as $\mathbf{F_r}$, where $\{\mathbf{F_r} \subseteq \mathbf{F}\}$.
T_{ij}	Positive-valued variable,T_{ij}, represents the traffic volume over the link e_{ij}.
Constant	**Definitions**
D_{th}	Time delay threshold value.
b_{ij}	Bandwidth of link e_{ij}.
R_f	$\begin{cases} 1, & \text{if flow } f \in \mathbf{F_r} \text{ is a real-time flow,} \\ 0, & \text{otherwise.} \end{cases}$
Decision Variable	**Definitions**
$FR(f,i,j)$	$\begin{cases} 1, & \text{if flow } f \in \mathbf{F} \text{ passes through link } e_{ij} \in E, \\ 0, & \text{otherwise.} \end{cases}$
L_{ij}	$\begin{cases} 1, & \text{if link } e_{ij} \in \mathbb{E} \text{ is active,} \\ 0, & \text{otherwise.} \end{cases}$

The packet rate of flow f is denoted λ_f. In order to place utility constraints on the solution, we introduce an inequality which states that the total packet rate of all flows passing through a link should not exceed the available bandwidth of that link, b_{ij} which is expressed as the difference between the bandwidth and the traffic volume, $b_{ij} - T_{ij}$. This is expressed as the inequality,

$$\sum_{f \in \mathbf{F}} FR(f,i,j) \cdot \lambda_f \leq b_{ij} - \mathrm{T}_{ij}, \quad \forall e_{ij} \in E. \tag{4}$$

A flow's packet rate is counted according to the directed nature of the network graph.

To ensure that every flow has a unique source $f.S_r$ and destination $f.D_s$, we introduce the following path conservation equalities,

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

We also introduce equalities that ensure a flow entering a node leaves the node for all intermediate nodes via the following flow conservation constraint,

$$\sum_{\substack{i=1 \\ i \neq f.S}}^{n} FR(f,i,j) = \sum_{\substack{i=1 \\ i \neq f.D}}^{n} FR(f,j,i), \quad \forall\, f \in \mathbf{F},\ j \in S. \tag{6}$$

We add constraints to ensure that the delay of an RTP flow does not exceed a user-determined QoS threshold. We define the delay, D_f, for the flow, f, to be the sum of delays experienced by the flow on the links it traverses,

$$D_f = \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} \cdot FR(f,i,j), \quad \forall f \in \mathbf{F}. \tag{7}$$

That inequalities that ensures that the delay of a RTP flow, D_f, does not exceed the QoS threshold, D_{th}, is expressed as

$$D_f \leq D_{\text{th}} \cdot R_f \quad \forall f \in \mathbf{F}. \tag{8}$$

In this constraint, if $R_f = 0$ for a flow f, it effectively means that the constraint is not applied to that flow. If $R_f = 1$, then the constraint ensures that the delay of the real-time flow does not exceed the QoS threshold, D_{th}.

To avoid network looping we add the following constraint,

$$FR(f,i,j) + FR(f,j,i) \leq 1, \quad \forall f \in \mathbf{F},\ i,j \in S. \tag{9}$$

Determining the optimal paths for network flows while ensuring that link capacities are not exceeded falls within the realm of the Multi-Commodity Flow (MCF) problem. This problem is well known for being Nondeterministic Polynomial-time-complete (NP-complete), particularly when dealing with integer flows, as established in [38]. In the experiments detailed in [10,37], the optimization process, utilizing leading solvers like Goroubi [39], LinGo [40], and set of OR-Tools from Google [41], typically takes several minutes to converge to an optimal solution, even for scenarios involving a relatively small number of flows, e.g., a few hundred flows. In conclusion, the computational demands of this InLP are impractical for an ideal online flow routing system. As a result, we resort to heuristic methods, which offers a more practical and efficient approach to developing power optimization algorithms.

5. Proposed Framework

The proposed framework is illustrated in Figure 4. It consists of three main layers: the infrastructure layer, the control layer, and the application layer. The infrastructure layer represents the network topology and includes traffic generators, VLC, and the Distributed Internet Traffic Generator (D-ITG), which are emulated within Mininet. This layer uses the secure channel provided by OpenFlow to communicate securely with the control layer. The control layer serves as the central hub for network information and management. The application layer is where the primary contributions of this framework are located. The three main contributions of this work are the Monitoring, Classifier, and Q-PoPS modules. We continue by describing the operation of each component within this framework and its contribution to the network architecture.

5.1. Traffic Generator

To simulate a realistic network workload, we adopted the open source tool D-ITG [42], which has seen widespread use. Using D-ITG, we can inject network traffic with multiple flows and exert packet-level control using a number of customized attributes such as volume, distribution, duration, start, and so on. D-ITG supports the client-server model. In our experimental topology, we implemented a D-ITG transmitter on the terminal of each traffic sender node and a D-ITG receiver on the terminal of each traffic receiver node. The characteristics of the generated traffic are summarized in Table 2.

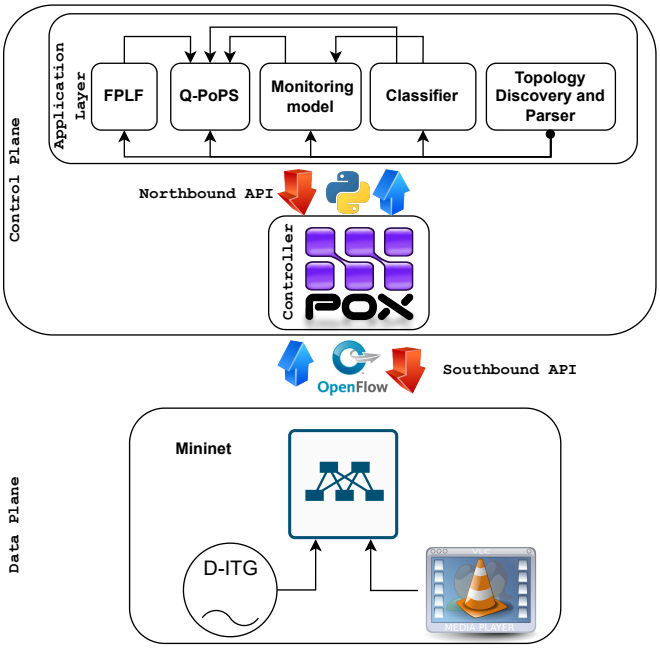


Figure 4. The Q-PoPS framework for power-optimized QoS for SD-DCNs is illustrated. It leverages the northbound (POX) and the southbound (OpenFlow) interfaces. The core innovation lies in Q-PoPS, which integrates information from the Classifier, Monitoring, and FPLF components to manage power and QoS for real-time traffic, which is represented by $\mathbf{F_r}$. Section 5.5 details the operation of Q-PoPS.

Table 2. Characteristics of the traffic used in the SD-DCN as outlined in [1].

Characteristics	Type of Distribution	Shape (α)	Scale (x_u)
OFF periods	Weibull	10	100
ON periods	Pareto	10	100

5.2. Topology Discovery and Parser Models

The SDN controller is responsible for network management. Management is achieved using the global view that the controller has of the entire network. To build a global view of the SDN, the network controller needs to discover all networking elements, such as switches, routers and links, that reside in its domain. This is conducted by incorporating the discovery https://github.com/noxrepo/pox (accessed on 22 July 2024) module of the POX controller in the current implementation of Q-PoPS framework. To facilitate the management of the collected data, the NetworkX [43] tool is used to build a graph model on top of the data plane layer. This graph is necessary for network representation and manipulation.

5.3. Classifier Model

To identify the RTP traffic, a real-time network monitoring technique empowered by flow-based classification is required. The authors in [30] introduced the SDN monitoring framework called Graph Modeling for OpenFlow Switch Monitoring (GMSM). GMSM has the capability of classifying the incoming traffic into different classes of service, i.e., RTP and non-RTP flows. We incorporate the GMSM classifier component into Q-PoPS. The classifier relies on the OpenFlow asynchronous `packet_in` and `flow_removed` messages to identify the class of service of the newly arrived packets. Once this is completed, the network graph is updated. Each link in the network is marked with a value between 1 and n, which corresponds to the number of services, or the number of flows passing over it. In practice, the output of the classifier is a Flow-Class Set, $\mathbf{F_c}$, where the flow set is

the set of all possible services, $\mathbf{F_c} = \{c_1, c_2, \ldots, c_n\}$ where examples of services include, $c \in \{icmp, video, voip \ldots\}$. Each link may map to one or more class of service. Each flow in the set is associated with a unique identifier, H_f, so that when the flow is terminated the controller will know which entry to remove from the flow table of the involved switches.

5.4. Monitoring Model

The tasks carried out by the monitoring model are listed as follows: (1) calculate the utilization for each link using the method presented in [10]; (2) monitor the delay of RTPA flows. One of the well-known approaches for checking QoS metrics is by using a probing packet. The probing packet can be used to calculate several important metrics, including the Round-Trip Time (RTT), number of hops, throughput between two points, as well as latency and its variation, which is commonly known as jitter. In this paper, we employ probing packets to calculate the flow delay time of each RTP flow in the SD-DCN during the execution of the power optimization algorithm, FPLF. Based on the delay measurement results we obtain, the proposed algorithm, Q-PoPS, which determines whether the system should find an alternative path for the RTP flow to maintain the delay of the RTPA within an acceptable level.

Implementing probing packets in SD-DCNs offers advantages such as centralized control, flexibility, and simplified management. We generate the probing packet in the controller concurrently with each emerging RTP event and send it to the first switch in the path. We have already saved all paths returned by the FPLF algorithm. The packet follows the flow's path. We instruct the last switch to send it back to the controller. It carries information in its payload which includes a hash identifier number, H_f, and the sending time. As a final stage, the controller extracts the information and matches its hash identifier, H_f, with the saved active flows to calculate the RTP flow delay. Consequently, the controller gains valuable insights into the flow's behavior, helping to identify potential issues, and aiding in optimizing the performance of the FPLF algorithm.

We break down the functions that we have described into three algorithms to complete our description. Algorithm 1 handles new and accomplished events, in network topology $G(S, E)$. In the first line, the algorithm initializes empty variables to store the sets of flows and paths. Subsequently, in lines 3–6, the algorithm handles each new flow event by identifying the efficient energy path using the FPLF function with a unique H_f and forwards it to the classifier for flow class verification. The algorithm proceeds to lines 7–8. In the case where an RTP flow is found, it activates the second algorithm, e.g., Algorithm 2. Line 11 of Algorithm 1 shows that when a flow completes, the corresponding key is deleted from the dictionary, **F**. The monitoring flow list is updated passively and periodically based on the switches that raised the event(s).

Algorithm 1 Flow event handling

Require: Network Topology $G(S, E)$, New/Accomplished Flow Events.
Ensure: Flow Set (**F**),

```
1:  F ← { }                                   ▷ Initialize the F dictionary.
2:  if New Flow Event then
3:      P ← FPLF(f)                           ▷ Find the energy efficient path.
4:      F[hashing(f)] = P
5:      Classifier (f)                        ▷ Call for the Classifier.
6:      if f is RTP then
7:          Call Algorithm 2
8:      end if
9:  end if
10: if Delete Flow Event then
11:     del F[hashing(f)]                     ▷ Delete the corresponding key from F.
12: end if
```

Algorithm 2 generates probing packets for the set of RTP flows in the SD-DCN. It takes the RTP flow set, $\mathbf{F_r} \subseteq \mathbf{F}$, as its input and sends a unique probing packet, $\mathtt{P}_{pkt}$, for each active flow, $f \in \mathbf{F_r}$. This procedure is described in lines 4–8. The probing packet is loaded with unique information, i.e., data including the current time and the unique hash ID, H_f. The algorithm then encapsulates the probing packet into an OpenFlow message and sends the message to the source switch of the flow. Finally, the algorithm is rescheduled for the next time it is run.

Algorithm 3 calculates the delays for different flows. It takes a probing packet, $\mathtt{P}_{pkt}$, as its input and determines the flow delay set, **D**, as its output. Line 1 of Algorithm 3 initializes an empty dictionary, **D**, which is used to store the flow delay set. Line 2 checks if a received packet matches the probing packet, $\mathtt{P}_{pkt}$. We use a private port number to distinguish it from traditional control plane traffic. In lines 3 to 6 of Algorithm 3, we decapsulate the payload from the packet and check if the payload matches any existing flow in the RTP flow dictionary, $\mathbf{F_r}$. If there is a match, it calculates the flow delay, D_f, by subtracting the departure timestamp ω_p from the arrival timestamp α_p, and then stores the flow delay D_f in the **D** dictionary, with the corresponding flow ID, H_f, as a key index. Line 8 ensures that if the payload does not match any flow in $\mathbf{F_r}$, the packet is ignored.

Algorithm 2 Probing packet generation

Require: Flow Set ($\mathbf{F_r}$),
Ensure: Probing Packet ($\mathtt{P}_{pkt}$)
1: `Monitoring` $\leftarrow$ `dict(`$\mathbf{F_r}$`)` ▷ Create a copy of the dictionary **F**.
2: **for** f `in Monitoring` **do**
3: $f_{src} \leftarrow f.S_r$
4: $\mathtt{P}_{pkt} \leftarrow$ `CreatPKT(·)` ▷ Generate a probing packet for analysis purposes, $\mathtt{P}_{pkt}$.
5: `Payload` $\leftarrow$ `CreatPL(·)` ▷ Form the payload, which includes the current time.
6: $\mathtt{P}_{pkt} \leftarrow$ `Encapsulation(Payload)` ▷ Pack information into the probing packet.
7: `msg` $\leftarrow$ `Encapsulation(`$\mathtt{P}_{pkt}$`)` ▷ Pack the probing packet $\mathtt{P}_{pkt}$ into the OF message.
8: `send(msg,`f_{src}`)` ▷ Send the message to the flow source switch, $f.S_r$.
9: **end for**
10: `scheduling(time, Algorithm 2)` ▷ Reschedule the function call.

Algorithm 3 Flow delay calculation

Require: Probing Packet ($\mathtt{P}_{pkt}$), Flow Set ($\mathbf{F_r}$)
Ensure: Flow delay set (**D**)
1: **D** $\leftarrow$ { } ▷ Create the dictionary, **D**.
2: **if** `New` P_{pkt} `Receiving Event`
3: payload = decapitation($\mathtt{P}_{pkt}$)
4: **if** payload.match in $\mathbf{F_r}$ **then** ▷ Retrieve the payload.
5: $D_f \leftarrow \alpha_p - \omega_p$ ▷ Calculate the delay.
6: **D**[payload.match] = D_F
7: **else**
8: pass
9: **end if then**
10: **end if**

We focus on monitoring active RTP flows in this paper. One of the advantages of the Q-PoPS approach is its lightweight nature. Another notable feature is its adaptability to dynamic network conditions. The Q-PoPS algorithms update the flow dictionaries **F**, $\mathbf{F_r}$, and $\mathbf{F_c}$ on an ongoing basis, based on network traffic patterns, i.e., adding new flow events and removing completed ones. This enables them to track and calculate delays for new flows that may emerge or for existing flows that undergo changes. By using the Q-PoPS classifier, the Q-PoPS approach can be extended to monitor a wider range of traffic types.

5.5. Q-PoPS Model

Figure 4 shows Q-PoPS obtains the final network state picture using inputs from the following model components: Classifier, Monitoring, and FPLF. Algorithm 4 defines the input and output of the Q-PoPS model. It iterates through all flows, f, belonging to the RTP flow set, $\mathbf{F_r}$, and checks the delay metrics for each flow. If the delay tolerance threshold is met, the Algorithm 4 proceeds. If not, Algorithm 4 calls FPLF to find an alternative path based on the current state of the SD-DCN and proactively installs it as an alternative path. Finally, it creates a new flow id, H_f, and updates all related dictionaries. The output solution of Algorithm 4 may either maintain the power of the SD-DCN at the same level or increase it to different values based on the state of the active subset of the topology.

Algorithm 4 Proactive path selection

Require: Network topology, $G(V, E)$, flow delay set, $\mathbf{D}$, RTP flow set, $\mathbf{F_r}$), and the flow set, $\mathbf{F}$.
Ensure: Alternative Path (AP)
1: **for** `each` $f \in \mathbf{F_r}$ **do**
2: $D_f \leftarrow \mathbf{D}[H_f]$ ▷ Check each flow delay.
3: **if** $D_f >$ `Time Delay Tolerance` **then**
4: $P \leftarrow$ FPLF(f) ▷ Find the energy efficient alternative path.
5: Proactive-Install(P) ▷ Install the path proactively.
6: $\mathbf{F}$[hashing(f)] = P ▷ Create a new flow id, H_f, for the new path, and update the dictionaries it belongs to.
7: $\mathbf{F_r}$[hashing(f)] = P
8: **end if**
9: **end for**

6. Experimental Results

In this section, we describe the experimental setup and present the results from our evaluation of the Q-PoPS framework. The duration of each experiment varies depending on the emulated scenario. We emulate a real-world fat-tree topology with $k = 4$, which is illustrated in Figure 5, using the Network Simulation Setup (FNSS) [44] and the Mininet Emulator [45]. The experiments involve injecting both high- and low-traffic loads into the emulated network to induce congestion using D-ITG. The packet sizes generated by D-ITG traffic range from 100 to 1500 bytes. The flow rate of traffic follows the characteristics outlined in [46], which are summarized in this paper, in Table 2. The RTP streaming traffic is sent from the terminal of each sender server using VLC player. The default VLC settings are used; these settings include the encoding and the streaming bitrate.

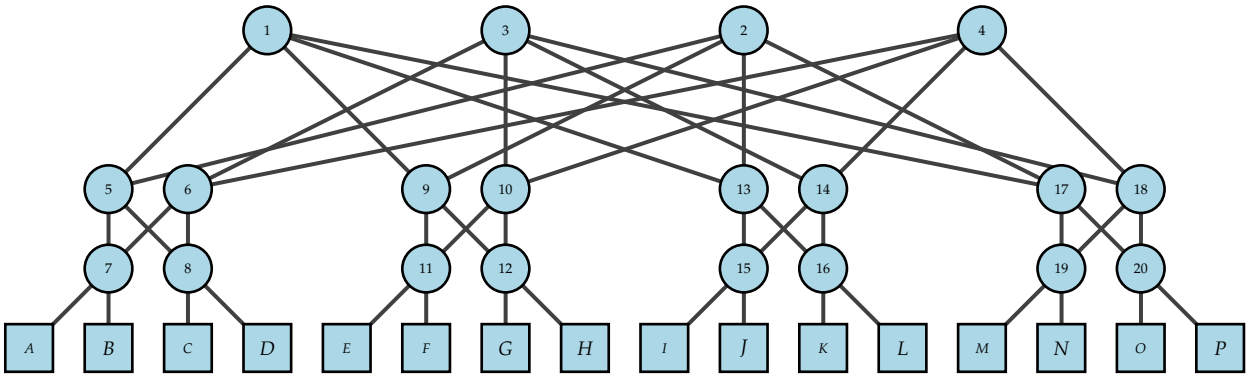


Figure 5. Emulated $k = 4$ fat-tree topology. The figure illustrates a fat-tree network where k represents the parameter that defines the number of switches and connections in the network. Specifically, $k = 4$ indicates a topology with four levels of switches and a total of $k^2 = 16$ switches in the network. This topology is often used in data center networks to provide scalable and fault-tolerant network structures.

6.1. Q-PoPS Experimental Scenarios

We subjected the Q-PoPS algorithm to a series of intensive experimental scenarios involving multi-flow transmission. These flows encompassed two types of flow: Internet Control Message Protocol (ICMP) flows and video streaming flows, which were transmitted between multiple source–destination pairs. The primary objective of this experiment was to assess the effectiveness of solutions generated by Q-PoPS in terms of the SD-DCN's power consumption and flow delay. Q-PoPS obtained three types of solution, where the solution obtained depended on the workload situation of the DCN.

To demonstrate the functionality of the proposed framework, we conducted an experiment involving the simultaneous transmission of multiple ICMP flows from server F to L and RTP flows from E and A to P and D, respectively. We recorded all the simulation details in a CSV file. A portion of this file is presented in Table 3. The second column displays the timestamp of the monitoring time of the flows. The third column shows the assigned flow ID, H_f, which highlights how each new flow obtains a unique flow ID.

The monitoring model utilizes these flow IDs to monitor each flow on an ongoing basis. The final column in the table represents the delay value, which is only calculated for RTP flows. Notably, the RTP flow with flow ID, $H_f = 4.94 \times 10^{18}$, experienced an increased delay due to the consolidation of multiple ICMP flows on one path. In contrast, the RTP flow with the flow ID, $H_f = -1.24 \times 10^{18}$ had an acceptable delay time, as the links' path remained under-utilized.

During these experiments, we observed that when link utilization exceeded 90%, the delay increased dramatically. Therefore, in this paper, we consider these events to indicate that the links were saturated.

Table 3. Flow delay analysis: We present a sample of the Q-PoPS flow delay analysis, focusing on monitoring traffic and its impact on delay, particularly for Real-time Transport Protocol (RTP) flows.

Index	Time	H_f	Flow Type	Flow Src and Dst	Path	Delay
1	17:10:20	4.94×10^{18}	RTP	E P	11 9 2 17 20	1.703501
2	17:10:29	-1.24×10^{18}	RTP	A D	7 5 8	1.057863
3	17:10:38	5.18×10^{18}	ICMP	F L	11 9 2 13 16	not applied
4	17:10:42	5.84×10^{18}	ICMP	F L	11 9 2 13 16	not applied
5	17:10:48	-6.55×10^{18}	ICMP	F L	11 9 2 13 16	not applied
6	17:10:58	1.57×10^{18}	ICMP	F L	11 9 2 13 16	not applied
7	17:11:08	-1.82×10^{18}	ICMP	F L	11 9 2 13 16	not applied
8	17:11:18	-1.59×10^{17}	ICMP	F L	11 9 2 13 16	not applied
9	17:11:28	-4.68×10^{18}	ICMP	F L	11 9 2 13 16	not applied
10	17:11:38	-3.00×10^{18}	ICMP	F L	11 9 2 13 16	not applied
11	17:11:48	-1.21×10^{18}	ICMP	F L	11 9 2 13 16	not applied
12	17:11:58	6.60×10^{18}	ICMP	F L	11 9 2 13 16	not applied
13	18:12:08	4.94×10^{18}	RTP	E P	11 9 2 17 20	161.152601
14	19:12:08	-1.24×10^{18}	RTP	A D	11 9 2 17 20	1.467466

6.1.1. Q-PoPS: Best-Case Results

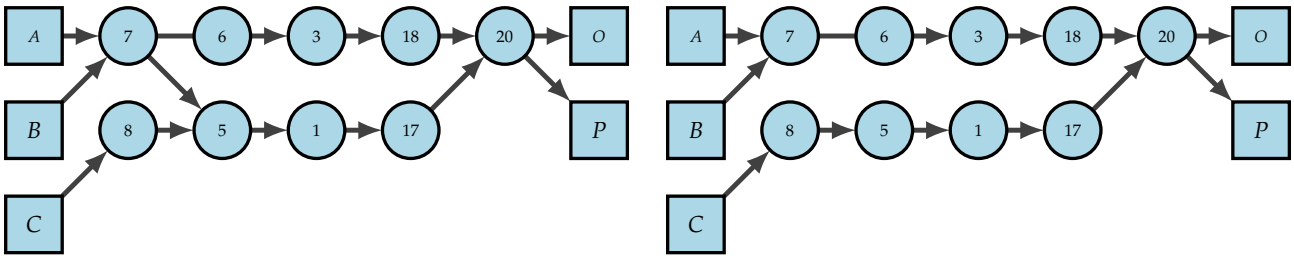
This experiment demonstrates that Q-PoPS can maintain power consumption at the same level whilst improving the QoS delay metric, when an alternative path meeting the QoS requirements is available. The events that occur during this scenario are summarized in Table 4. The simulation commenced at time 11:56:20 (hours:minutes:seconds). We gradually injected multiple ICMP traffic flows from server B to destination O, via the path $(7, 6, 3, 18, 20)$. All the sources and destinations of flows are shown in Figure 5. This continued until time 11:58:30 when the path entered a saturated or over-utilized state.

Table 4. Paths and saturation states as a function of simulation time: The state of paths during periods of saturation and their impact on delay is described.

Start Time	Path	Number of Flows	Saturated Period (s)	Start/End of Saturated Period	Delay of Saturated Period
11:56:20	$(7,6,3,18,20)$	multiple ICMP	120	11:58:30 to 12:00:30	not applied
11:58:20	$(7,5,1,17,20)$	single RTP	60	12:01:00 to 12:02:00	delay ≥ 75
11:59:10	$(8,5,1,17,20)$	multiple ICMP	60	12:01:00 to 12:02:00	not applied
12:01:00	$(7,6,3,18,20)$	multiple ICMP/single RTP	none	none	delay ≤ 75

Subsequently, we initiated new RTP and ICMP traffic flows from servers A and C to the destination P. This event prompted FPLF to establish new paths, $(7,5,1,17,20)$ and $(8,5,1,17,20)$, which shared the $(5,1,17,20)$ segment of the path, to optimize power consumption. We then incrementally increased the traffic load until the new paths entered a saturated state. This occurred at time 12:01:00, and resulted in the delay exceeding the threshold value of $D_{\text{th}} = 75$ ms, which we applied in this experiment. In summary, the delay event was triggered as a result of the impact of the delay on RTP flow quality.

The active subset that optimized power consumption before and after the delay event is shown in Figure 6. Q-PoPS reverted to the topology's active subset, which is illustrated in Figure 6a, to search for a path meeting the RTP requirement. Some of the ICMP flows originating from server B ended, and the links in the path $(7,6,3,18,20)$ became under-utilized. They had an acceptable delay time; the utilization was lower than 90%. Q-PoPS redirected the RTP flow to the selected path, generating a new flow ID, H_f, which connected to the current state of the flow without increasing power consumption costs. The final state of the graph as a consequence of these actions is shown in Figure 6b.



(**a**) (**b**)

Figure 6. Illustration of the active subset of the network graph, G, before (LHS window) and after (RHS window) a delay event triggered by congested real-time traffic. (**a**) Q-PoPS: active subset of the graph, G, before the delay event. (**b**) Q-PoPS, active subset of the graph, G, after the delay event.

The right panel of Figure 7 shows that the power consumption of the SD-DCNs is the same for both the FPLF and the Q-PoPS scenarios. This occurs despite Q-PoPS being aware of the QoS delay metric. This is consistent with the results in [47] which state that an OpenFlow switch consumes approximately 0.2 W/port when it operates at 10 Mbps. In both cases, the SD-DCN consumed the same amount of power, 3.6 Watts.

The left panel of Figure 7 illustrates how we monitored the delay value of the RTP flow using its flow ID, H_f, both before and after the delay event trigger. At time 12:01:00, the flow ID, H_f of the RTP flow changed as Q-PoPS assigned a new path with a new flow ID, H_f. This makes sense; the graph aligns with the result in Figure 8, which shows the link utilization of active path link plots as a function of simulation time, specifically for the shared segment $(5,1,17,20)$. This segment is a component of both paths, $(7,5,1,17,20)$ and $(8,5,1,17,20)$, and it reached saturation at time 12:01:00, causing a dramatic increase in delay, so that the delay exceeded the specified threshold of 75 ms in Figure 7. In closing, note that the ratio of active ports before and after the event, e.g., $P_r = \frac{active-ports(after)}{active-ports(before)}$, is 1.

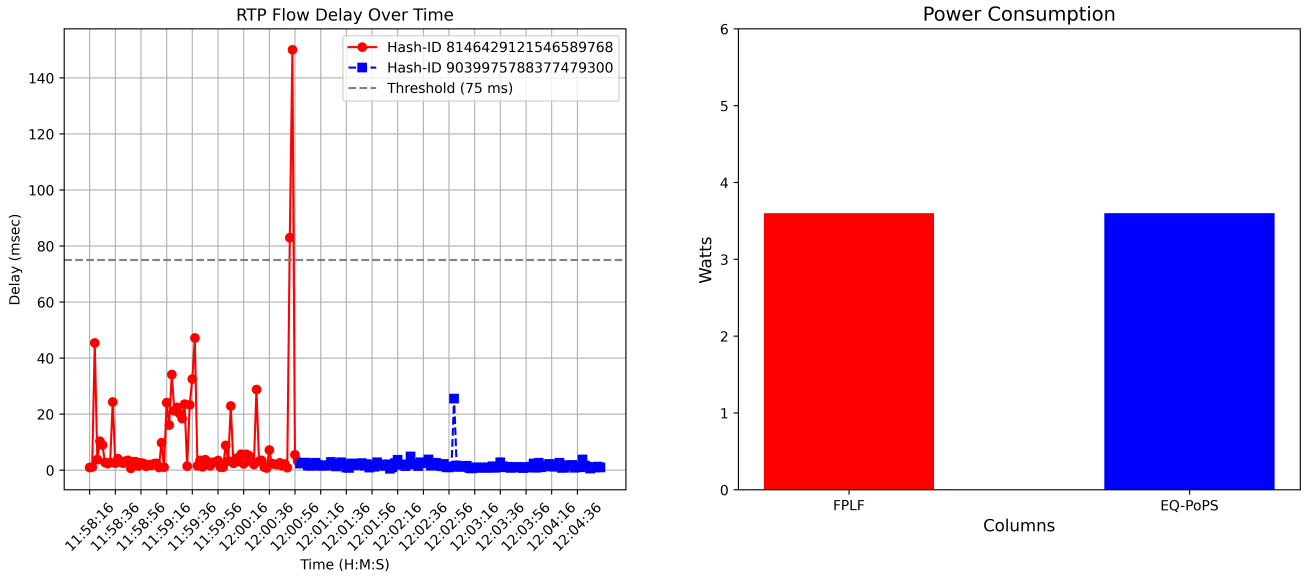


Figure 7. Comparison of power consumption, using the pure FPLF algorithm proposed in [10], and the Q-PoPS method (RHS window), along with delay value of the RTP Flow before and after the delay event (LHS window).

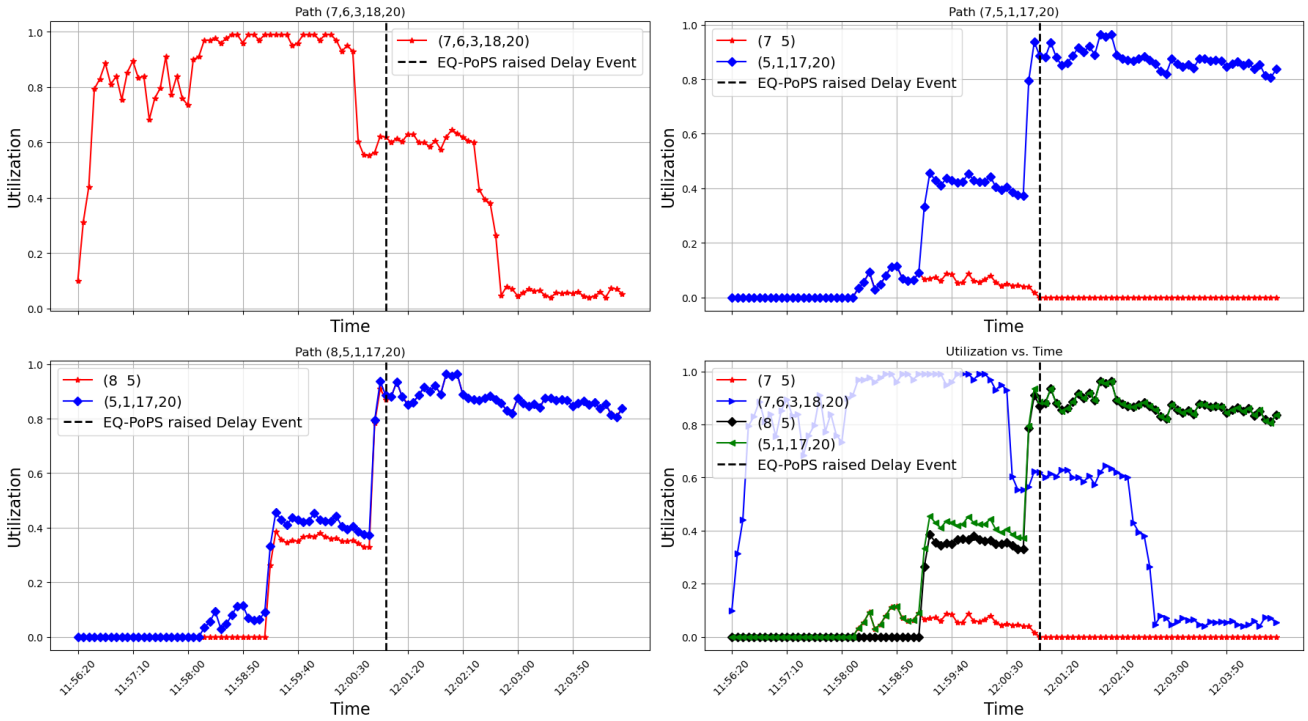


Figure 8. Visualization of the active link path utilization as a function of simulation time. The upper LHS window displays the utilization for the path $(7, 6, 3, 18, 20)$, where all the links in the path have the same utilization. In the upper RHS window, the path $(7, 5, 1, 17, 20)$ is shown, where all links have the same utilization except for link, $(7, 5)$, because it is not a shared link. The utilization of path $(8, 5, 3, 17, 20)$ is presented in the lower LHS window. The lower RHS window displays all the critical links that Q-PoPS relied on to make decisions about alternative paths.

6.1.2. Q-PoPS: Midrange-Case Results

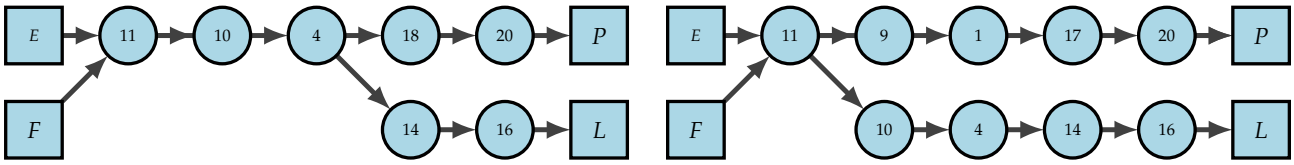
Q-PoPS offers another solution involving the replacement of bottleneck links in a path experiencing delay. This is illustrated in the following scenario. Table 5 shows that the simulation commenced at time 11:20:00 with the injection of RTP traffic from server

E to server P over the path $(11, 10, 4, 18, 20)$. Subsequently, we injected ICMP traffic from server F to server L. FPLF responded by routing the traffic to the path $(11, 10, 4, 14, 16)$. We incrementally increased the traffic load until the segment path $(11, 10, 4)$ became saturated with over-utilized links at time 11:22:05. At this simulation time, a delay event occurred, prompting Q-PoPS to seek an alternative path for the RTP flow.

Table 5. Paths and saturation states as a function of simulation time: In response to congestion changes as time evolves, Q-PoPS finds alternative paths.

Start Time	Path	Number of Flows	Saturated Period (s)	Start/End of Saturated Period	Delay of Saturated Period
11:20:00	$(11, 10.4, 18, 20)$	single RTP	120	11:22:05 to 11:24:25	delay ≥ 75
11:20:10	$(11, 10, 4, 14, 16)$	multiple ICMP	120	11:22:05 to 11:24:25	not applied
11:22:20	$(11, 9, 1, 17, 20)$	single RTP	none	none	delay ≤ 75
11:22:20	$(19, 17, 20)$	multiple ICMP	none	none	not applied

The active subset of the topology before and after the event is shown in Figure 9. Q-PoPS reverted to the topology's active subset, which is illustrated in Figure 9a, to search for a path meeting the RTP requirement. Leveraging the similarity of the DCN topology, particularly the fat-tree architecture, Q-PoPS rerouted the RTP flow to the other core switch connected to the same POD within the topology, e.g., switch 1. This transition occurred via the alternative path $(11, 9, 1, 17, 20)$, which was proactively installed with a new flow ID, H_f, satisfying the RTP flow requirements between servers E and P. In this scenario, Q-PoPS deactivated the links in the subpath $(4, 18, 20)$, rendering them idle. They were replaced by the newly activated links on the subpath $(1, 17, 20)$ which was connected to core switch 1. Consequently, Q-PoPS reduced power consumption by utilizing only four ports instead of the original eight ports—using a totally new path—while maintaining low delay with a slight increase in DCN power consumption. This change is shown in Figure 9b.



(**a**) (**b**)

Figure 9. We illustrate how Q-PoPS optimizes power consumption for the midrange traffic scenario by showing the state of the graph, G, before and after the delay event. (**a**) Q-PoPS: active subset of graph, G, before the delay event. (**b**) Q-PoPS: active subset of graph, G, after the delay event.

To demonstrate Q-PoPS' capability to consolidate more flows with the RTP flow after rerouting it to a new path, we initiated multiple ICMP flow bursts from server N to server O. As the new alternative path remained under-utilized, the FPLF approach rerouted the ICMP flows over the path $(19, 17, 20)$ to increase its utilization, specifically targeting the shared link $(17, 20)$, as shown in the last row of Table 5.

The right panel of Figure 10 reveals a slight difference between the power consumption of the SD-DCNs when FPLF and Q-PoPS are used. Notably, the FPLF approach consumes less power than Q-PoPS. This is because Q-PoPS switched ON 4 additional ports, resulting in a power consumption of 3.2 Watts, as opposed to the 2.4 Watts consumed by FPLF. The left panel demonstrates how Q-PoPS dynamically assigns a new path with a new flow ID, H_f, for the RTP flow after the delay event was triggered. This occurred at time 11:20:05.

All the actions taken by Q-PoPS are closely linked to the utilization of active path links during the simulation, as depicted in Figure 11. It can be observed that the shared links $(11, 10, 4)$ reach saturation at time 11:22:05. Concurrently, the utilization of the link $(17, 20)$ increases after the delay event. This particular link becomes a shared resource, accommodating two types of flows, ICMP and RTP flow, which originate from different

sources and are destined for different destinations. In contrast, the links $(4, 18, 20)$ go into an idle state after the delay event. This allows them to contribute to the reduction in power consumption. As a closing point, we calculate the ratio of active ports, P_r, before and after the event, and determine that it is 1.5.

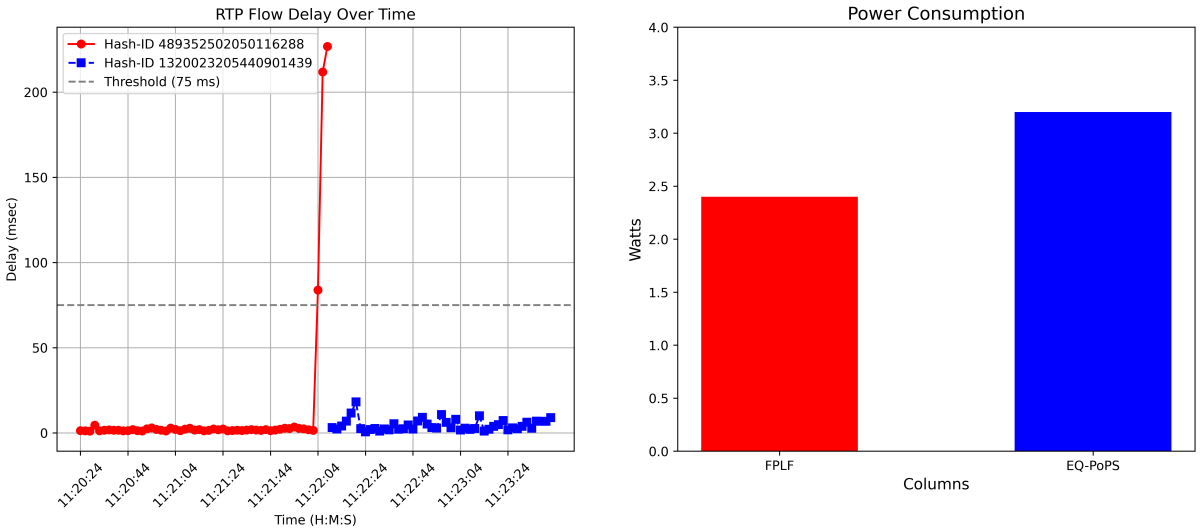


Figure 10. We compare the power consumption of SD-DCNs managed by FPLF and Q-PoPS. FPLF exhibits lower power usage (2.4 Watts). Q-PoPS consumes a larger power consumption (3.2 Watts). Q-PoPS dynamically reroutes the real-time traffic flow when congestion occurs, requiring the activation of additional ports, 4 in this case. The primary result communicated is the trade-off between power efficiency and real-time traffic management.

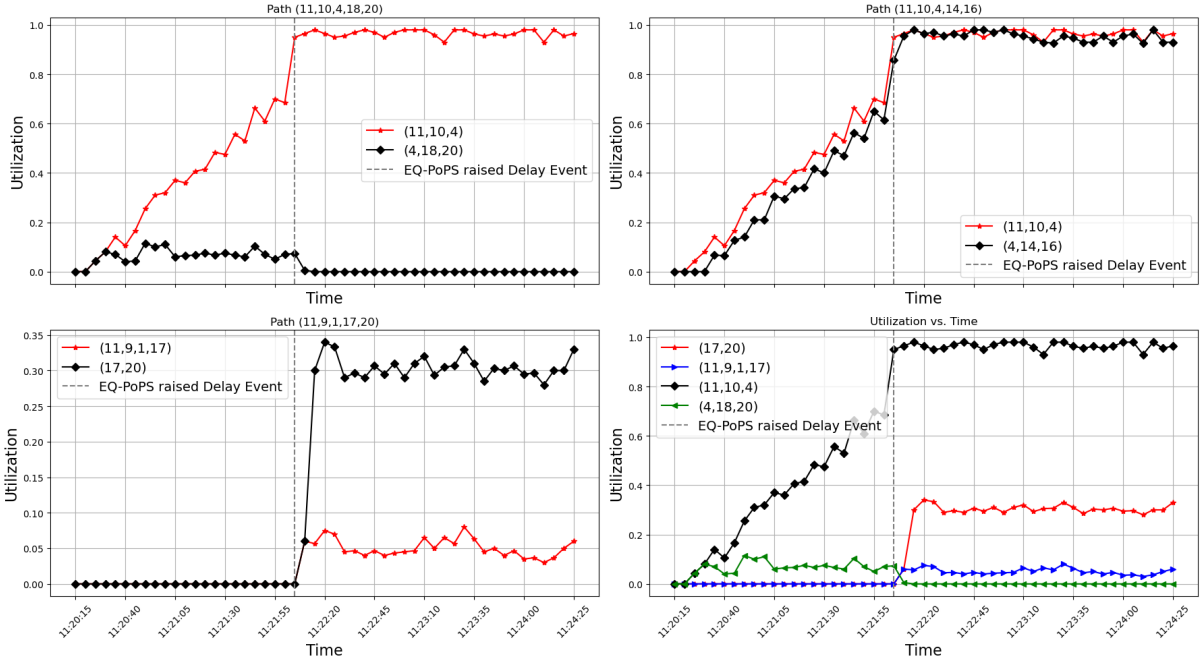


Figure 11. Visualization of active link path utilization as a function of simulation time. The upper LHS window shows the link utilization of the path $(11, 10, 4, 18, 20)$ as time evolves. The upper RHS second displays the path link utilization for the path $(11, 10, 4, 14, 16)$. The path segment $(11, 10, 4)$ is utilized more because it supports both ICMP and RTP traffic before the delay event. Conversely, the link path $(4, 14, 16)$ carries only ICMP throughout the simulation. The lower LHS window presents the utilization of the link path $(11, 9, 1, 17, 20)$. The lower RHS window showcases all the critical links that Q-PoPS relies on for decision-making regarding alternative paths.

6.1.3. Q-PoPS: Worst-Case Results

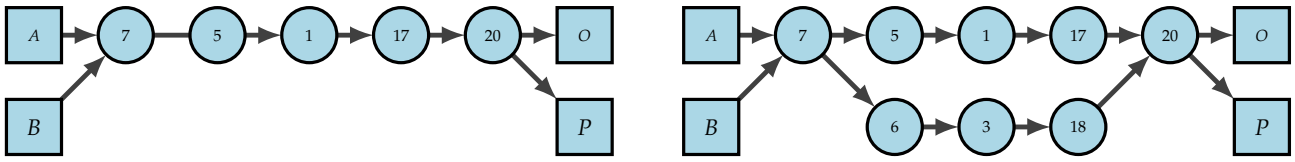
If Q-PoPS exhausts the above options, it resorts to the solution described in this subsection, which involves finding a new shortest path to reroute the RTP flow. A scenario demonstrating this is described below. The path states at various times are summarized in Table 6.

Table 6. Path states in the worst case scenario simulation as a function of simulation time.

Start Time	Path	Number of Flows	Saturated Period (s)	Start/End of Saturated Period	Delay of Saturated Period
17:01:11	$(7, 5, 1, 17, 20)$	single RTP/multiple ICMP	140	11:04:20 to 11:06:20	delay ≥ 75
17:04:01	$(7, 6, 3, 18, 20)$	single RTP	none	none	delay ≤ 75

The simulation commenced at time 17:01:11 with the injection of RTP traffic from server B to server P over the path $(7, 5, 1, 17, 20)$. Subsequently, we injected ICMP traffic between source, A, and destination, O. Since the links involved in the path $(7, 5, 1, 17, 20)$ were still under-utilized, FPLF merged the new flow with the existing RTP flow. We then incrementally increased the traffic load until the entire path $(7, 5, 1, 17, 20)$ became saturated with over-utilized links. This occurred at time 17:04:00 of the simulation. At this point in the simulation, a delay event occurred, prompting Q-PoPS to search for an alternative path for the RTP flow.

The active subset of the topology before and after the event is displayed in Figure 12. As there is only one active path, Q-PoPS cannot make use of the topology's active subset. This is illustrated in Figure 12a. Therefore, it found an entirely new shortest path, $(7, 6, 3, 18, 20)$, with a new flow ID, H_f, to meet the RTP flow requirements between servers B and P. As a result, Q-PoPS doubled the power consumption by utilizing an additional eight ports with the original eight ports. This action yielded a ratio of active ports, before and after the delay event, of $P_r = 2$. This action was taken to obtain a solution for the worst-case scenario that improved the delay metric using traffic consolidation techniques to optimize power usage. The resulting topology is shown in Figure 12b.



(**a**) (**b**)

Figure 12. Q-PoPS optimizes delay in the worst-case scenario; however, due to a limited active subset (**a**), Q-PoPS finds a completely new path (**b**) to meet real-time traffic needs. (**a**) Q-PoPS: active subset of graph G before the delay event. (**b**) Q-PoPS: active subset of a graph, G, after the delay event.

Figure 13 compares the power consumption of Q-PoPS (5 Watts) and FPLF (2.5 Watts). Additionally, it illustrates Q-PoPS' monitoring of RTP delay using the flow ID, H_f, before and after an event. At time 17:04:00, Q-PoPS assigned a new path with a new flow ID, H_f. The utilization of the old path $(7, 5, 1, 17, 20)$ saturated, resulting in an increase in delay that exceeded the threshold of 75 ms. The new path $(7, 6, 3, 18, 20)$ gradually increased around the same time, 17:04:00, indicating an improvement in the RTP flow. The link utilization of active path links over the simulation time is shown in Figure 14.

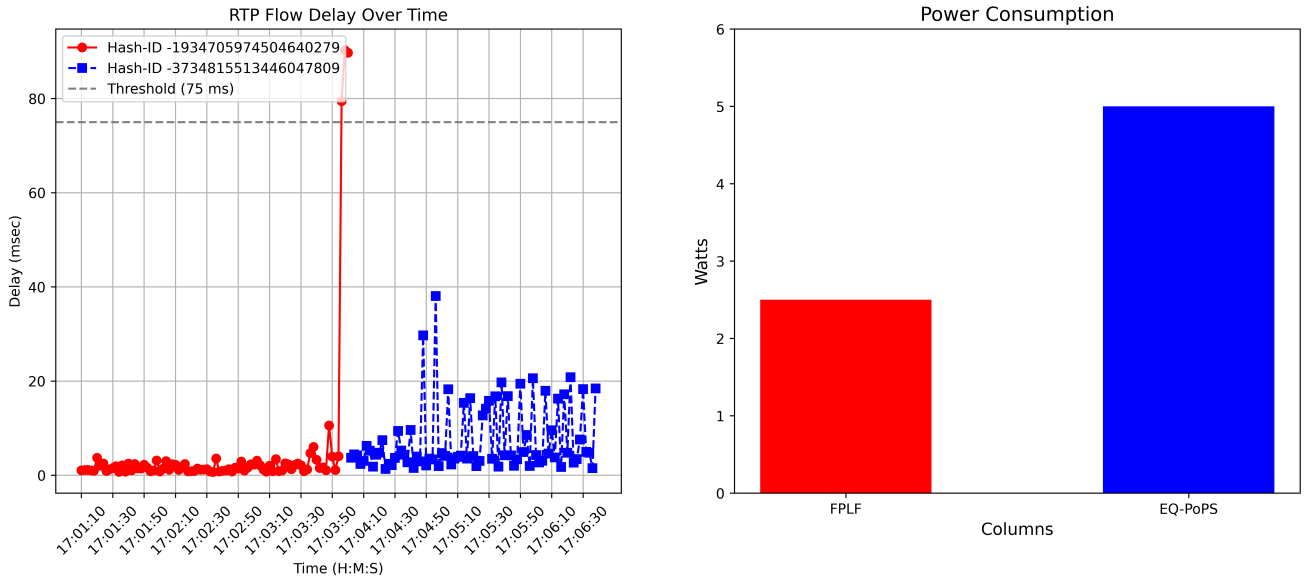


Figure 13. Comparison of power consumption between FPLF and Q-PoPS (RHS window), along with delay value for the RTP Flow before and after a delay event (LHS window), worst-case solution.

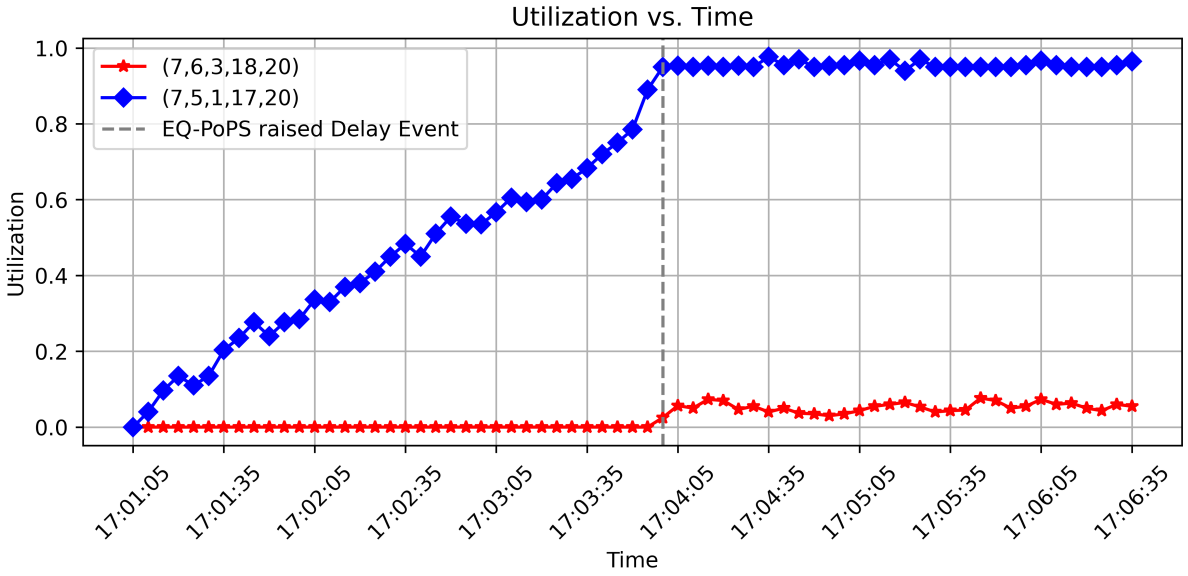


Figure 14. Visualization of paths utilization during the simulation. The path $(7,5,1,17,20)$ shared both ICMP and RTP traffic until 17:04:00, then it started carrying ICMP traffic only for the remaining simulation time, while the path $(7,6,3,18,20)$ served as the alternative path for transmitting the RTP flow.

7. Discussion and Future Work

We have demonstrated Q-PoPS' ability to monitor RTPA flows in SD-DCNs using unique flow IDs, H_f. It manages network congestion effectively by dynamically redirecting flows to less congested active paths. This was evident in the best-case results obtained in the experiments section, where Q-PoPS maintained the active ports ratio at 1, optimizing the use of shared resources and reusing the available active ports. In the midrange-case, Q-PoPS increased the active ports ratio to be in the range, $1 < P_r < 2$, by replacing the bottleneck part of the path.

In cases where all links in the active subset become overloaded, Q-PoPS increased energy consumption as the last possible resort. It achieved this by finding a completely

new path with the same or more hops, which increased the ratio of active ports to $P_r \leq 2$, thus maintaining the network's quality.

In our future work, we plan to conduct further research and testing in larger and more diverse SD-DCN scenarios. This will enable us to determine if Q-PoPS can handle the challenges posed by multiple parallel flow traffic effectively and to improve its performance under these demanding conditions. Additionally, we aim to optimize power consumption by proactively reconsolidating large flows after the completion of short-flow transmissions, which can lead to increased power savings over time. Finally, ongoing efforts to enhance the algorithm's runtime and scalability will remain a priority for future improvements.

8. Conclusions

In recent years, there has been a growing concern about the trade-off between power consumption and QoS metrics in SD-DCNs. This issue is crucial for both subscribers and service providers. In this paper, we have focused on the rising significance of RTP flows in SD-DCNs, which require special treatment due to their unique characteristics.

The existing approach, FPLF, primarily aims to minimize power consumption. To address the challenge posed by RTP traffic, we introduced a new Integer Programming model and heuristic algorithm named Q-PoPS. Q-PoPS is specifically designed to optimize power consumption while handling the requirements of RTP traffic in a real-time manner.

We conducted simulations under three different congestion scenarios, comparing the performance of FPLF and Q-PoPS. The Q-PoPS algorithm provided solutions across three scenarios: best-case, midrange-case, and worst-case, focusing on power optimization while considering flow delay metrics.

In the best-case scenario, both Q-PoPS and FPLF demonstrated similar power consumption levels. The ratio of active ports, P_r, is 1. This indicates that the introduction of Q-PoPS did not increase power consumption ratios while effectively maintaining low delay for real-time traffic.

In the midrange- and worst-case scenarios, Q-PoPS made adjustments by activating additional ports to reroute real-time traffic during congestion, thereby sacrificing some power consumption savings. The ratio of active ports, P_r, is 1.5 in the midrange scenario and 2 in the worst-case scenario. This proactive approach effectively prevented delays, maintaining latency under 75 ms, contrasting with FPLF, which prioritized power savings but often resulted in increased delays.

Finally, the paper shows the importance of considering the class of service of different applications in modern SD-DCNs while also focusing on power optimization. In future work, we intend to explore the validity of Q-PoPS outside of DCNs and under diverse workload conditions and scales.

Author Contributions: Conceptualization, M.N.; Formal analysis, G.K.; Funding acquisition, R.d.F.; Investigation, G.K. and R.d.F.; Methodology, M.N., A.M. and R.d.F.; Project administration, R.d.F.; Resources, A.M.; Software, M.N. and A.M.; Supervision, G.K.; Validation, G.K. and R.d.F.; Writing—original draft, M.N. and A.M.; Writing—review editing, R.d.F. All authors have read and agreed to the published version of the manuscript.

Funding: This paper has emanated from research supported by a Grant from Science Foundation Ireland under Grant numbers 13/RC/2077_P2 and 15/SIRG/3459.

Data Availability Statement: The software models generated and the tools used by this research will be made available to the public on GitHub repositories [48].

Acknowledgments: The authors would like to express their sincere gratitude to Science Foundation Ireland for their generous support of this research. The authors would also like to thank the Department of Information Technology at the University of Debrecen for providing them with the necessary resources and facilities to carry out this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

DCNs	Data Center Networks
VoIP	Voice over IP
QoS	Quality of Service
Q-PoPS	QoS-Aware Power-Optimized Path Selection
DVFS	Dynamic Voltage and Frequency Scaling
SNMP	Simple Network Management Protocol
SDN	Software-Defined Networking
FA	Flow Aggregation
FS	Flow Scheduling
ECMP	Equal Cost Multi-Path
InLP	Integer Linear Monogramming
DRL	Deep Reinforcement Learning
PoD	Point of Delivery
EEP	Energy-Efficient Paths
MDP	Markov Decision Process
IoT	Internet of Things
RTPA	Real-Time Protocol Applications
TM	Traffic Matrix
LU	Link-Utility
LC	Link-Cost
FSP	Fill Shortest Path
CM	Cost Matrix
MCF	Multi-Commodity Flow
D-ITG	Distributed Internet Traffic Generator
FNSS	Network Simulation Setup

References

1. Nsaif, M.; Kovásznai, G.; Malik, A.; de Fréin, R. SM-FPLF: Link-State Prediction for Software-Defined DCN Power Optimization. *IEEE Access* **2024**, *12*, 79496–79518. [CrossRef]
2. Assefa, B.G.; Özkasap, O. A survey of energy efficiency in SDN: Software-based methods and optimization models. *J. Netw. Comput. Appl.* **2019**, *137*, 127–143. [CrossRef]
3. Rios, A.; Gonzalez, A.J.; Alcober, J.; Ozon, J.; Ghafoor, K.Z. Conferencing Services in P2P Networks: Trends and Challenges. In *Future Internet Services and Service Architectures*; River Publishers: Aalborg, Denmark, 2022; pp. 139–160.
4. Masanet, E.; Shehabi, A.; Lei, N.; Smith, S.; Koomey, J. Recalibrating global data center energy-use estimates. *Science* **2020**, *367*, 984–986. [CrossRef]
5. Hintemann, R.; Hinterholzer, S. Energy consumption of data centers worldwide: How will the Internet become green? In Proceedings of the ICT4S, Lappeenranta, Finland, 10–14 June 2019; Volume 5.
6. Bari, M.F.; Boutaba, R.; Esteves, R.; Granville, L.Z.; Podlesny, M.; Rabbani, M.G.; Zhang, Q.; Zhani, M.F. Data center network virtualization: A survey. *IEEE Commun. Surv. Tutor.* **2012**, *15*, 909–928. [CrossRef]
7. Internet Engineering Task Force (IETF). IP Flow Information Export (IPFIX) Protocol Specification. Technical Report, RFC 7011. 2013. Available online: https://datatracker.ietf.org/doc/html/rfc7011 (accessed on 22 July 2024).
8. Cisco Systems, Inc. NetFlow Services and Solutions Guide. 2008. Available online: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/15-mt/nf-15-mt-book/get-start-cfg-nflow.html (accessed on 22 July 2024).
9. Guo, Z.; Hui, S.; Xu, Y.; Chao, H.J. Dynamic flow scheduling for power-efficient data center networks. In Proceedings of the IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), Beijing, China, 20–21 June 2016; pp. 1–10. [CrossRef]
10. 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. [CrossRef]
11. Pranata, A.A.; Jun, T.S.; Kim, D.S. Overhead reduction scheme for SDN-based data center networks. *Comput. Stand. Interfaces* **2019**, *63*, 1–15. [CrossRef]
12. Cisco. Global Cloud Index Projects Cloud Traffic to Represent 95 Percent of Total Data Center Traffic by 2021. 2018. Available online: https://newsroom.cisco.com/press-release-content?articleId=1908858 (accessed on 22 July 2024).
13. da Silva Conterato, M.; Coelho Ferreto, T.; Rossi, F.; dos Santos Marques, W.; Silas Severo de Souza, P. Reducing energy consumption in SDN-based data center networks through flow consolidation strategies. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 1384–1391. [CrossRef]

14. Wang, X.; Yao, Y.; Wang, X.; Lu, K.; Cao, Q. Carpo: Correlation-aware power optimization in data center networks. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 1125–1133. [CrossRef]
15. 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 NSDI, San Jose, CA, USA, 28–30 April 2010; Volume 10, pp. 249–264.
16. Assefa, B.G.; Ozkasap, O. Framework for Traffic Proportional Energy Efficiency in Software Defined Networks. In Proceedings of the IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Batumi, Georgia, 4–7 June 2018; pp. 1–5. [CrossRef]
17. Lu, Z.; Lei, J.; He, Y.; Li, Z.; Deng, S.; Gao, X. Energy optimization for Software-Defined data center networks based on flow allocation strategies. *Electronics* **2019**, *8*, 1014. [CrossRef]
18. Hossain, M.M.; Georges, J.P.; Rondeau, E.; Divoux, T. Energy, carbon and renewable energy: Candidate metrics for green-aware routing? *Sensors* **2019**, *19*, 2901. [CrossRef]
19. Zafar, S.; Ahmad Chaudhry, S.; Kiran, S. Adaptive trimtree: Green data center networks through resource consolidation, selective connectedness and energy proportional computing. *Energies* **2016**, *9*, 797. [CrossRef]
20. Yao, Z.; Wang, Y.; Qiu, X. DQN-based energy-efficient routing algorithm in software-defined data centers. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720935775. [CrossRef]
21. HongYu, P.; FuJian, S.; Kan, W.; TianLu, H.; DeQuan, X.; LeXi, X. A Non-Cooperative Data Center Energy Consumption Optimization Strategy Based on SDN Structure. In Proceedings of the 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, Shenyang, China, 20–22 October 2021; pp. 1386–1390. [CrossRef]
22. Luo, J.; Zhang, S.; Yin, L.; Guo, Y. Dynamic flow scheduling for power optimization of data center networks. In Proceedings of the Fifth International Conference on Advanced Cloud and Big Data (CBD), IEEE, Shanghai, China, 13–16 August 2017; pp. 57–62. [CrossRef]
23. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic flow scheduling for data center networks. In Proceedings of the NSDI, San Jose, CA, USA, 28–30 April 2010; Volume 10:8 , pp. 89–92.
24. Li, D.; Yu, Y.; He, W.; Zheng, K.; He, B. Willow: Saving data center network energy for network-limited flows. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 2610–2620. [CrossRef]
25. Xu, G.; Dai, B.; Huang, B.; Yang, J. Bandwidth-aware energy efficient routing with SDN in data center networks. In Proceedings of the 17th International Conference on High Performance Computing and Communications, 7th International Symposium on Cyberspace Safety and Security, and 12th International Conference on Embedded Software and Systems, IEEE, New York, NY, USA, 24–26 August 2015; pp. 766–771. [CrossRef]
26. Xu, G.; Dai, B.; Huang, B.; Yang, J.; Wen, S. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. *Future Gener. Comput. Syst.* **2017**, *68*, 163–174. [CrossRef]
27. Lozano-Rizk, J.E.; Gonzalez-Trejo, J.E.; Rivera-Rodriguez, R.; Tchernykh, A.; Villarreal-Reyes, S.; Galaviz-Mosqueda, A. Application-aware flow forwarding service for sdn-based data centers. *Electronics* **2022**, *11*, 3882. [CrossRef]
28. Wang, S.; Karmoshi, S.; Saleh, F.; Alhusaini, N.; Li, J.; Zhu, M.; Hawbani, A. GANA-VDC: Application-aware with bandwidth guarantee in cloud datacenters. *Electronics* **2019**, *8*, 258. [CrossRef]
29. Tsai, P.W.; Tsai, C.W.; Hsu, C.W.; Yang, C.S. Network monitoring in software-defined networking: A review. *IEEE Syst. J.* **2018**, *12*, 3958–3969. [CrossRef]
30. Malik, A.; de Fréin, R. Graph Modeling for OpenFlow Switch Monitoring. *IEEE Access* **2023**, *11*, 84543–84553. [CrossRef]
31. Tootoonchian, A.; Ghobadi, M.; Ganjali, Y. OpenTM: Traffic matrix estimator for OpenFlow networks. In *Passive and Active Network Measurement*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 201–210. [CrossRef]
32. Yu, C.; Lumezanu, C.; Zhang, Y.; Singh, V.; Jiang, G.; Madhyastha, H.V. Flowsense: Monitoring network utilization with zero measurement cost. In *Passive and Active Measurement (PAM)*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 31–41. [CrossRef]
33. Lisas, T.; de Fréin, R. Sequential Learning for Modeling Video Quality of Delivery Metrics. *IEEE Access* **2023**, *11*, 107783–107797. [CrossRef]
34. Lisas, T.; de Fréin, R. Quantum Classifiers for Video Quality Delivery. In Proceedings of the 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Gothenburg, Sweden, 6–9 June 2023; pp. 448–453. [CrossRef]
35. Nsaif, M.; Kovásznai, G.; Abboosh, M.; Malik, A.; de Fréin, R. ML-Based Online Traffic Classification for SDNs. In Proceedings of the 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), Debrecen, Hungary, 16–18 May 2022; pp. 217–222. [CrossRef]
36. Nsaif, M.; Kovásznai, G.; Malik, A.; de Fréin, R. Survey of Routing Techniques-Based Optimization of Energy Consumption in SD-DCN. *Infocommunications J.* **2023**, *15*, 35–42. [CrossRef]
37. Kovásznai, G.; Nsaif, M. Integer Programming Based Optimization of Power Consumption for Data Center Networks. *Acta Cybernetica* **2023**. [CrossRef]
38. Even, S.; Itai, A.; Shamir, A. On the complexity of time table and multi-commodity flow problems. In Proceedings of the 16th Annual Symposium on Foundations of Computer Science (SFCS), Berkeley, CA, USA, 13–15 October 1975; pp. 184–193. [CrossRef]
39. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2022. Available online: https://www.gurobi.com/documentation/9.5/refman/ (accessed on 22 July 2024).

40. LINDO Systems Inc. LINGO the Modeling Language and Optimizer. 2020. Available online: http://www.lindo.com (accessed on 14 September 2023).
41. Perron, L.; Furnon, V. OR-Tools. 2019. Available online: https://developers.google.com/optimization/ (accessed on 14 September 2023).
42. Botta, A.; Dainotti, A.; Pescapé, A. A tool for the generation of realistic network workload for emerging networking scenarios. *Comput. Netw.* **2012**, *56*, 3531–3547. [CrossRef]
43. Hagberg, A.; Swart, P.; S Chult, D. *Exploring Network Structure, Dynamics, and Function Using NetworkX*; Technical Report; Los Alamos National Lab. (LANL): Los Alamos, NM, USA, 2008.
44. 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, ICST, Brussels, Belgium, 5–7 March 2013; SIMUTOOLS.
45. 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, Monterey, CA, USA, 20–21 October 2010; pp. 1–6. [CrossRef]
46. Benson, T.; Anand, A.; Akella, A.; Zhang, M. Understanding data center traffic characteristics. *ACM SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 92–99. [CrossRef]
47. 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), Rio de Janeiro, Brazil, 17–21 November 2014; pp. 181–186.
48. Nsaif, M. QoS-Aware Power-Optimized Path Selection (Q-PoPS). 2024. Available online: https://github.com/nsaif86/Q_PoPS (accessed on 2 June 2024).