Short thesis for the degree of doctor of philosophy (Ph.D)

Sustainable Networking: A Study on Power Optimization in DCNs Using Traffic-Aware Methods, Along with SD-DCNs Emulation Strategies

by Mohammed Al-shamarti

Supervisor: Dr. Gergely Kovásznai

UNIVERSITY OF DEBRECEN
Doctoral School of Informatics
Debrecen, 2024

Contents

1 Introduction

DCNs have gained increasing prominence in our daily lives due to the rapid growth of modern information technologies, such as the Internet of Things, Big Data, Cloud Computing, and Mobile Sensing Networks [24, 23]. Therefore, DCNs are designed with a focus on high reliability and stability, achieved through the implementation of *redundant links* and *sufficient capacity*.
Thus, the motivation for achieving energy-efficient usage of DCNs is closely linked to the increasing digitization of daily life. The demand for digital services is growing at a remarkable pace. Since 2010, the number of internet users worldwide has doubled, and global internet traffic has expanded twentyfold. For example, according to information from Statista[1] [27], in 2020, data generation and replication reached an unprecedented peak, totaling 64.2 zettabytes, exceeding earlier predictions. This surge can be attributed primarily to the heightened demand resulting from the COVID-19 pandemic, with more individuals working and studying from home and engaging in home entertainment activities more frequently. Looking ahead, it is projected that global data production will exceed 180 zettabytes in the five years leading up to 2025. The same report indicated that only a small fraction of the newly generated data is preserved for the long term. Specifically, just about two percent (2%) of the data produced and used in 2020 was retained into 2021. Reflecting the substantial growth in data volume, the total storage capacity in data centers is expected to surge, with a projected Compound Annual Growth Rate (CAGR) of 19.2 percent between 2020 and 2025. By the end of 2020, the total installed storage capacity had already reached a significant 6.7 zettabytes. However, the proliferation of

[1] https://www.statista.com/statistics/871513/worldwide-data-created

data centers and transmission networks that support digitalization has resulted in increased energy consumption. This has two significant impacts (1) higher costs for both of service providers and end-users[2] [6], as well as (2) environmental concerns related to Carbon Dioxide (CO2) emissions[3] [1, 25]. 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 of energy demand from data centers and data transmission networks. Estimates indicate that in 2005, data centers consumed 153 TWh, and by 2010, this consumption had increased to 273 TWh of global electricity use, accounting for approximately $1 - 1.5\%$ of global electricity demand [16]. Furthermore, according to a research note from the Borderstep Institute, it is projected that the energy consumption of data centers worldwide may increase by a factor of 40 through 2030 [10].

On the other hand, a notable issue arises as network devices typically operate at *full capacity* 24 hours a day, resulting in substantial energy consumption. Unfortunately, these network devices are *underutilized* most of the time, leading to extremely low network energy efficiency [20]. Consequently, this problem has captured the attention of many researchers who are working to develop techniques that can save energy while still maintaining network performance, such as in [1].

Therefore, researchers, in collaboration with strong government and industry efforts on energy efficiency and renewable energy procurement, play a crucial role in mitigating and curbing energy demand and emissions growth over the next decade.

[2]Currently, Europe's energy grid is facing an unparalleled crisis. Since early 2021, wholesale costs of electricity and gas have increased by as much as 15 times, which has had devastating consequences on both individuals and companies.

[3]Currently, large-scale alternatives to fossil fuels that are secure, affordable, and low-carbon are lacking globally. The connection between access to energy and greenhouse gas emissions is the aspect of energy that receives the most attention

1.1 Thesis Points

In this section, we summarized the key theses derived from the dissertation, categorized under three main areas: ILP (Integer Linear Programming) for power usage modeling, heuristic algorithms for real-time emulation, and machine learning algorithms for mitigating the effects of consolidation techniques.

1.1.1 Thesis 1: Power Usage Modeling in DCNs Using a Consolidation ILP Model

1. **Problem Statement**: The escalating power usage in DCNs is a critical concern. Traditional network structures face significant limitations in optimizing power consumption, necessitating advanced methodologies.

2. **Approach**: This thesis focuses on solving the power optimization problem in DCNs using ILP formulations.

3. **Key Findings**: The ILP model was implemented and solved using various ILP solvers, including LinGo, GUROBI, CP-SAT, SCIP, and CBC. Comprehensive experiments were conducted on different traffic patterns (near, long, and random) using the tool NEO-DCN. The results showed that for the near traffic pattern, GUROBI and CP-SAT outperformed other solvers in terms of runtime. The long traffic pattern highlighted the challenges posed by increasing traffic volumes, with CP-SAT maintaining stability even in high-demand scenarios. The random traffic matrix experiment further emphasized the superior performance of GUROBI in terms of both runtime and memory consumption across a diverse range of traffic instances. Overall, the findings suggest that GUROBI is a robust and efficient solver for the power optimization problem in DCNs, providing valuable insights for future research and practical implementations.

1.1.2 Thesis 2: Real-time Emulation to Efficient Power Usage in SDN Environment

1. **Problem Statement**: Efficient power usage in SDN environments requires real-time solutions that can adapt to varying traffic conditions while maintaining performance.

2. **Approach**: This thesis explores the emulation setup and implementation of the Fill Preferred Link First (FPLF) algorithm in software-defined data center networks.

3. **Key Findings**: The FPLF algorithm was tested on a fat-tree topology using Mininet and the POX controller. Experiments demonstrated that in high-traffic scenarios, the FPLF algorithm achieved up to 10% energy savings, while in low-traffic situations, the algorithm realized an impressive 63.3% energy savings compared to the existing ECMP routing algorithm. These results underscore the algorithm's adaptability and significant energy savings in various traffic scenarios. However, the FPLF algorithm does have limitations, particularly in its ability to estimate flows accurately and equitably distribute them. Additionally, it does not prioritize specific types of traffic, such as VoIP, video streaming, and Real-Time Protocol Applications (RTPA), which can suffer from increased delays when mixed with other traffic types. This limitation highlights the need for further enhancements in flow estimation models and the development of mechanisms to ensure equitable flow distribution and priority handling of real-time traffic.

1.1.3 Thesis 3: Mitigation of the Effects of Consolidation Technique on the Quality of the RTPA Classes

1. **Problem Statement**: FPLF can blindly balance traffic flows between the core switches under high traffic conditions without considering the specific services of the flow classes. However, certain classes, such as VoIP, video streaming,

and Real-Time Protocol Applications (RTPA), can suffer from increased delays when mixed with other types of traffic. Consequently, such an algorithm could potentially impact the performance of real-time protocol applications. This realization prompted us to prioritize services for these specific classes, necessitating the development of a robust monitoring system capable of effectively identifying and rerouting such traffic to paths that meet their requirements. The central component of this monitoring system is the **traffic classifier**, which plays a pivotal role in enhancing the performance of SD-DCNs. Therefore, this thesis proposes learning algorithms to classify a wide range of traffic in SDN environments.

2. **Approach**: This thesis leverages machine learning models for traffic classification within SDN to enable distributed flow according to class type.

3. **Key Findings**: Three machine learning models were implemented: Logistic Regression, Support Vector Machine, and Neural Network. These models were trained and assessed using both synthetic and real traffic datasets. The Neural Network model emerged as a standout performer, exhibiting high accuracy in classifying diverse traffic types, up to 89%. Further integration of the Neural Network model into an SDN environment for real-time online testing indicated commendable accuracy across various traffic classes, though challenges were noted in classifying specific types such as games and telnet. The findings suggest that the proposed machine learning approach shows promising potential for optimizing power consumption within SD-DCNs.

2 Thesis 1: Power Usage Modeling in DCNs Using a Consolidation ILP Model

Flow Consolidation techniques aggregates data flows into a smaller set of links and switches that are sufficient to support existing data traffic demands, subject to a tolerance to a certain level of delay, packets loss, etc. To achieve minimum power consumption for a specific traffic matrix, switches and ports that are being used unnecessarily are put into sleep or shutdown mode. The flows could share one link fairly based on the Transmission Control Protocol (TCP) sharing scheme. The disadvantage of these techniques is that using only a subset of the switches and links, a sub-topology, may result in performance degradation, which is typically characterized by a QoS measure. This QoS measure may indicate the significance of increases in delay time (i.e., due to computational complexity of the output solutions), or the extent to which links with higher utilization become overloaded and more susceptible to unplanned failures [15]. Balancing between the level of energy consumption and routing techniques that meet a desired QoS is of great importance. Next, we introduce and discuss the concept of proposed model.

2.1 Problem Modeling

Power consumption of DCN routing mechanisms depends on the mode of operation of the set of switches that forward the flows between sources and destinations. Power consumption of DCN switches can be measured in two ways: *dynamic*, which measures the power consumption of active links, and *static*, which measures the sum of power consumed by components such as chassis, fans, and switching fabric. We adopt the following notation conventions, all the notations are summarized in Table 2.1. A directed weighted graph, $\mathbb{G} = (\mathbb{S}, \mathbb{E})$, models the network topology. The vertex-set is denoted by $\mathbb{S} = \{s_1, s_2, \ldots, s_n\}$, and the edge-set is denoted by $\mathbb{E} \subseteq \{e_{ij} \mid s_i, s_j \in \mathbb{S}\}$. The i-th switch is denoted as s_i and represents an OpenFlow switch. The primary function of each switch is to facilitate the routing of information through the path determined by the network controller. In the graph, $\mathbb{G}$, every edge represents a link, and the link connecting the i-th and j-th switches is identified by e_{ij}. Network links can exist in either an active (ON) or inactive (OFF) state. We use binary variables, denoted by L_{ij}, to indicate the current state of network links. The variable L_{ij} is 1 if the link connecting switches i and j is active. This means that it can transmit packets between two ports. Conversely, L_{ij} is 0 if the link is inactive. In practice, each link consists of two ports, a sending port and a receiving port. Therefore, when designing power-efficient routing the working ports of each link should be considered. Similarly, the variable, ℓ_i, is set to 1 if the switch is active and 0 if the switch is inactive. The Network Power Consumption (NPC) is given in (2.1).

$$\text{NPC} = S_p \sum_{s_i \in \mathbb{S}} \ell_i + D_p \sum_{e_{ij} \in \mathbb{E}} L_{ij}. \tag{2.1}$$

Equation (2.1) relates the NPC to ℓ_i and L_{ij}, which denote the state of a corresponding switch, s_i, and link, e_{ij}, i.e., whether they are turned on or off. The variables D_p and S_p denote whether the power consumption is dynamic or static, respectively, they are represented as non-negative real numbers, including zero,

$D_p, S_p \in \mathbb{R}_{\geq 0}$. In order to capture the flows traversing the topology, the variable $FR\,(f, i, j)$ takes the value 1, which means that the flow f traverses the edge e_{ij}. If e_{ij} is not traversed by the flow f, $FR(f, i, j)$ it 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, e_{ij}. The traffic of the DCN is represented by the set of flows $\mathbb{F}$, where each $f \in \mathbb{F}$ is defined as $f = \left(f.S, f.D, \lambda_f\right)$. 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 $f.S$ and $f.D$, respectively. Finally, the packet rate of flow f is denoted λ_f and is measured in bits per second.

2.1.1 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, $P \in \mathbb{P}$. 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.

3. Splitting a single flow into packets across multiple links in the topology could save energy by increasing overall link

Table 2.1: Conventional names of parameters used in problem formulation.

Parameters	**Definitions**
$\mathbb{S}$	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.
$f.S$	Source switch.
$f.D$	Destination switch.
$\mathbb{F}$	Set of flows, where a flow $f = \left(f.S, f.D, \lambda_f\right) \in \mathbb{F}$ is represented by source $f.S \in \mathbb{S}$, destination $f.D \in \mathbb{S}$, and bit rate $\lambda_f \in \mathbb{N}$.
$\mathbb{P}$	Set of energy-efficient shortest paths, where a path, $P = \{s'_1, \ldots, s'_k\} \in \mathbb{P}$ is represented by a set of switches between the servers of DCNs, where $\forall s'_i \in \mathbb{S}$.
$\mathbf{U}$	Utilization matrix where U_{ij} represents the utilization of the link e_{ij}.
Constant	**Definitions**
BW_{ij}	Adjacency matrix scaled by the bandwidth of the edges $\mathbb{E}$ equal to 1 Gbps.
$T_{ij} \in \mathbb{N}$	Positive-valued variable representing the traffic volume over e_{ij} before including the new flow (previous load state).
Decision Variable	**Definitions**
$FR\,(f, i, j)$	$\begin{cases} 1, & \text{if flow } f \in \mathbf{F} \text{ passes through link } e_{ij} \in \mathbb{E} \\ 0, & \text{otherwise} \end{cases}$
L_{ij}	$\begin{cases} 1, & \text{if link } e_{ij} \in \mathbb{E} \text{ is active} \\ 0, & \text{otherwise} \end{cases}$

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

$$\text{minimize} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} \right). \tag{2.2}$$

The objective function in Equation (2.2) 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. The above objective function works against the following constrains:

1. **Links and Traffic Correlation Constraint:** This constraint considers the correlation between the traffic volume and the links. Therefore, the constraint defines the relationship between the traffic volume T_{ij} and the link state L_{ij} to increase the utility of the link as much as possible, as shown in (2.3).

$$\frac{T_{ij}}{BW_{ij}} \leq L_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \tag{2.3}$$

   Expressing that the traffic volume must not exceed the bandwidth of a link.

2. **Links and Flows Correlation Constraint:** This constraint represents the correlation between links and flows, such that a link should be active if and only if a flow passes through it, as shown in (2.4).

$$FR\,(f,i,j) \leq L_{ij}, \quad \forall f \in \mathbb{F},\ \forall e_{ij} \in \mathbb{E}, \tag{2.4}$$

Meaning that flows should pass only through active links.

3. **Utility Constraint:** This constraint computes the utility of all the topology's links, and limits the link utility to less than or equal to the link's bandwidth BW_{ij}, thus this constraint acts to limit the value of U_{ij} (the left-hand side of the constraint), by considering the active flows f_{ij} and their packet rate values λ_f, as shown in (2.5).

$$\sum_{f \in \mathbb{F}} FR\,(f, i, j) \cdot \lambda_f \leq BW_{ij} - \mathrm{T}_{ij}, \quad \forall e_{ij} \in \mathbb{E}, \tag{2.5}$$

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

4. **Path Conservation Constraint:** This constraint installs the path from the source $f.S$ to the destination $f.D$ for each flow f, as shown in (2.6).

$$\sum_{i=1}^{n} FR\,(f, f.S, i) = 1, \quad \sum_{i=1}^{n} FR\,(f, i, f.D) = 1, \quad \forall f \in \mathbb{F}. \tag{2.6}$$

5. **Flow Conservation Constraint:** This constraint guarantees for any flow f that the number of the incoming and outgoing flows of the intermediate switches between the source $f.S$ and the destination $f.D$ should be equal, in order to avoid packet loss, as shown in (2.7).

$$\begin{aligned} \sum_{i=1}^{n} FR(f, i, j) = \sum_{i=1}^{n} FR(f, j, i), \\ \forall f \in \mathbb{F},\ \forall j \in \mathbb{S},\ j \neq f.S,\ j \neq f.D \end{aligned} \tag{2.7}$$

6. **Network Loop Avoidance Constraint:** In order to enhance the computational performance for finding the minimum number of links between the source switch, $f.S$, and destination switch $f.S$, we introduce the following constraint. This constraint aims to prevent nodes from revisiting their

immediate predecessors in the path, as shown in (2.8).

$$FR(f,i,j) + FR(f,j,i) \leq 1, \quad \forall f \in \mathbb{F},\ i,j \in \mathbb{S}. \tag{2.8}$$

2.2 Problem Solving

We evaluated the proposed model using various solvers including: LinGo, which is a high-level language for optimization modeling; GUROBI [9], which provides one of the most powerful commercial solvers for a wide range of optimization problems, including integer programming (IntP) problems, and is free to use for academics and students; CP-SAT[1], which is a constraint programming solver that uses SAT methods and is part of the OR-TOOLS package; SCIP [4], one of the fastest non-commercial solvers for mixed integer programming (MIntP) and an open-source framework for constraint integer programming; and CBC (Coin-or Branch and Cut), an open-source MIntP solver [8].
We run GUROBI, CP-SAT, SCIP, and CBC from Python code inside our portfolio solver NEO-DCN[2], which is publicly available at https://github.com/kovasz/neO-DCN, while LinGo is used as a standalone platform on Windows.
NEO-DCN runs the mentioned solvers in versatile combinations on our IntP model to find the optimal solution. The source code and benchmarks are publicly available[3], fostering further research in energy-efficient networking. This chapter reports on experiments under three communication patterns (near, long, and random), measuring runtime and memory consumption to evaluate the performance of different IntP solvers.

2.3 Experiment Setup

In our experiments, we are dealing with the real-world DCN topology, fat-tree [22]. The topology containing 20 switches and

[1] https://developers.google.com/optimization/cp/cp_solver
[2] https://github.com/kovasz/neO-DCN
[3] https://github.com/kovasz/neO-DCN

16 hosts, and numerous links between those nodes. The bandwidth of each link is uniformly set to 1 Gbps.
The IntP solvers were run on the benchmark instances with a wall clock time limit of 1200 seconds. LinGo was run as a Windows desktop application, while the other IntP solvers as part of NEO-DCN on Linux. During the experiments, we measured the runtime of the solvers and, also, monitored the memory consumption of solvers by using the memory profiler `mprof`. While applying `mprof` to NEO-DCN was successful, we were not able to apply it to LinGo. This is why we will not provide memory consumption data for LinGo.

2.4 Result Discussion

The comparative experiments have been conducted on a wide range of traffic benchmarks for three different communication patterns: (1) the near traffic pattern results show that GUROBI and CP-SAT outperform the other solvers regarding runtime for most traffic instances; (2) the long traffic pattern results show that above a traffic volume of 2.7 Gbps all the solvers dramatically loose efficiency, except for CP-SAT, which keeps a good performance and roughly constant memory consumption; (3) the random traffic results show that, for most of the traffic instances, GUROBI outperforms the other solvers regarding both runtime and memory. We can conclude that, for most of the benchmark instances, most of the solvers outperform LinGo regarding runtime. Consequently, it was definitely worth experimenting with those solvers, with GUROBI and CP-SAT in particular, as part of our new contribution in [13]. Figure 2.1 shows the results for the long traffic pattern case.

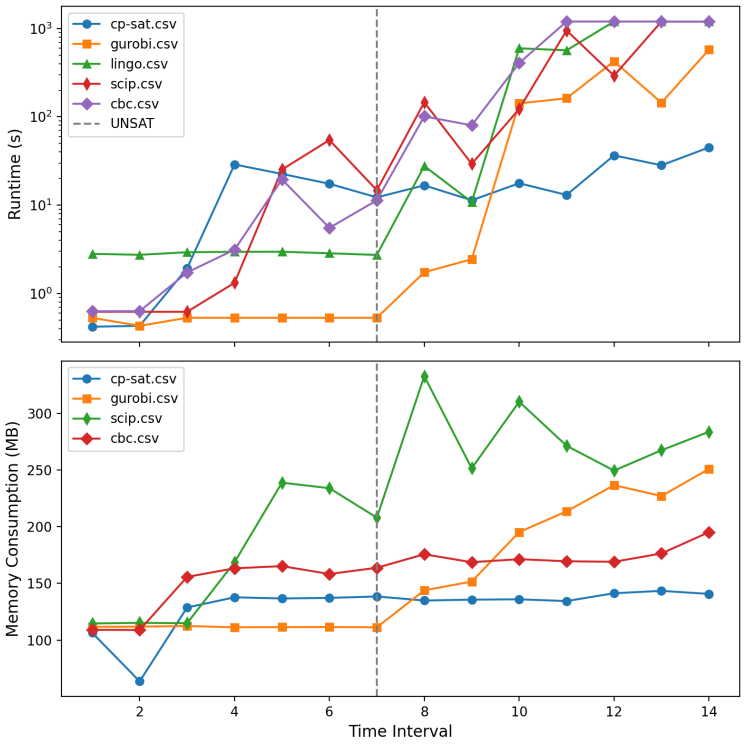

Figure 2.1: The upper plot shows the runtimes of ILP solvers for the long traffic pattern (note that time out was set to 1200 seconds), while the lower plot displays the memory consumption of ILP solvers for the same pattern.

3 Thesis 2: Real-time Emulation to Efficient Power Usage in SDN Environment

Building upon the findings in Section 2, the real-time implementation and emulation of our idea have become crucial for its practical application in an industrial context. In this chapter, we introduce a novel framework designed to minimize power consumption in software-defined DCNs. Central to this framework is the Fill Preferred Link First (FPLF) algorithm, a heuristic link utility-based approach that seeks to strive a balance between energy efficiency and performance.

FPLF is specifically designed to maintain QoS and optimize the energy consumption of DCNs. This is accomplished through continuous monitoring of DCN traffic conditions, utilizing the OpenFlow protocol [18] to acquire real-time topology state and data traffic information. The algorithm selects the most energy-efficient path below a pre-defined threshold, ensuring effective flow quality and power-aware routing.

We demonstrate the effectiveness of FPLF in maintaining satisfactory QoS while reducing power consumption in DCNs, as evidenced by the count of dropped packets, and latency. Therefore, we evaluate our proposed framework using an experimental platform comprising the POX controller and the Mininet network emulator.

Finally, we benchmark our proposed algorithm against the state-of-the-art approach, the Equal Cost Multi-Path (ECMP) algorithm, to provide a comprehensive comparison and assessment. It is worth noting that a substantial portion of the content in this chapter draws upon the insights presented in the work of [21]. All the notations used in the chapter are presented in Table 3.1.

Table 3.1: Conventional names of parameters used in problem formulation.

Parameters	Definitions
$\mathbb{S}$	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.
f	A flow $f = \left(f.S, f.D, \lambda_f\right) \in \mathbb{F}$ represented by source, destination and packet rate.
$\mathbb{P}$	Set of energy-efficient shortest paths, where a path, $P = \{s'_1, \ldots, s'_k\} \in \mathbb{P}$ is represented by a set of switches between the servers of DCNs, where $\forall s'_i \in \mathbb{S}$.
$\mathbf{U}$	Utilization matrix where U_{ij} represents the utilization of the link e_{ij}.
$\mathbf{C}$	Link-cost matrix where C_{ij} denotes the cost of the link e_{ij}.
$\mathbf{A}$	Aggregated traffic matrix.
$\mathbf{T}$	Traffic matrix where T_{ij} denotes input traffic of e_{ij}.
BW_{ij}	Adjacency matrix scaled by the bandwidth of the edges $\mathbb{E}$.
k	Number of the pods in the fat-tree topology.
s_r	Source switch.
d_s	Destination switch.

3.1 Proposed Method

Similar to the proposed IntP Objective Function discussed in Section 2.1.1, our FPLF algorithm aims to minimize the energy consumption of the DCN. This objective is achieved by maximizing the link utilization, as expressed in (3.1), which is designed to aggregate traffic as efficiently as possible across a minimum number of links:

$$\max \left(\frac{\sum_{e_{ij} \in \mathbb{E}} T_{ij}}{BW^{th}} \right), \quad \forall e_{ij}. \tag{3.1}$$

The FPLF algorithm ensures that all switches utilize the specified link as long as its bandwidth is underutilized. However, if the link exceeds the threshold value of 90% of the bandwidth, the FPLF algorithm redirects flows to the shortest alternative path using a Dijkstra algorithm.

Before the FPLF algorithm starts, it is essential to have comprehensive statistics on traffic within the DCN. To address this requirement, we developed a robust monitoring model.

3.1.1 Monitoring Model

To ensure the effective functioning of FPLF, we have developed a monitoring model as a prerequisite. This model is designed to collect real-time statistics, specifically the traffic from each node, utilizing the OpenFlow protocol 1.0. The OpenFlow protocol facilitates the exchange of extensive statistics, such as flow, port, and group table statistics, through messages between the SDN controller and OpenFlow switches. These statistics provide the controller with a comprehensive global view of the DCN.

In our scenario, where each link consists of two ports, we collect port statistics by periodically sending OpenFlow "port stats request" messages to all switches. Upon receiving the response messages, the monitoring model extracts the bytes and packets cumulative values.

All counters in OpenFlow switches, including Port Counters, Flow Counters, Group Counters, and others, typically maintain

cumulative values. These values represent the total count or amount of a specific metric since the counter was last reset or initialized. Consequently, the collection matrix at any given point in time serves as an aggregated traffic matrix. In our case, the aggregated traffic matrix, $\mathbf{A}$, is generated during the monitoring phase. Figure 3.1 illustrates how this matrix is generated.
At time t, the controller issues a polling statistics message, and the aggregated traffic represented by $\mathbf{A}(t)$ is temporarily stored until the subsequent time frame $t+1$, when a new traffic aggregated traffic matrix, $\mathbf{A}(t+1)$, is constructed. The controller then accurately calculates the traffic matrix, $\mathbf{T}$, for a specific time frame, as shown in (3.2):

$$\mathbf{T} = \frac{\mathbf{A}(t) - \mathbf{A}(t-1)}{\Delta t} \tag{3.2}$$

Where Δt is the polling period, which is always one second in our case. Hence, we use $\mathbf{T}$ to calculate the link utilization matrix, $\mathbf{U}(t)$, for each period.

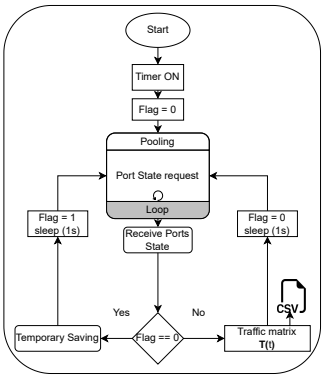

Figure 3.1: Procedure of Traffic Matrix generation.

3.1.2 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 $f = (f.S, f.D, \lambda_f)$ passes through a link, e_{ij}, its corresponding utility U_{ij} is increased by $\frac{\lambda_f}{BW_{ij}}$.

Algorithm 1: Link-Utility.

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

```
1 U = 0
2 foreach edge e_ij ∈ G do
3 |   foreach (f.S, f.D, λ_f) ∈ F passing through e_ij do
4 |   |   U_ij ← U_ij + λ_f / BW_ij
5 |   end
6 end
```

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.

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

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, $\left(f.S, f.D, \lambda_f\right)$. The input parameters are the network graph, $\mathbb{G}$, the link-cost matrix, **C**, and the incoming flows $\left(f.S, f.D, \lambda_f\right)$. 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, (overload, idle, and 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 $O\left(|\mathbb{S}|^2\right)$ complexity as it depends on Dijkstra algorithm to pick paths based on the minimum number of hops.
As a conclusion, the FPLF algorithm utilizes three attraction components, as depicted in Figure 3.2, to generate a network topology subset that satisfies the current traffic demand. Simultaneously,

Algorithm 3: Fill-Shortest Path.

Input : edge set $\mathbb{E}$, flow $\left(f.S, f.D, , \lambda_f\right)$, link-cost matrix **C**
Output: energy-saving path *ESP*
foreach *new request* **do**
    Run Dijkstra's algorithm [7] to find the shortest path with minimum link costs (C_{ij})
end

it places all other nodes in the DCN in an OFF state for power saving.

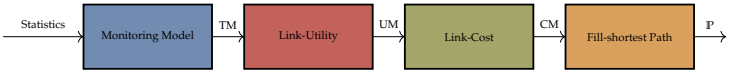

Figure 3.2: The FPLF algorithm's components are interconnected, with the output of each component serving as the input to the subsequent one. The culmination of this process yields the Energy-Efficient Shortest Path as the final output.

3.2 Proposed Framework

The experimental platform is constructed using the Mininet network emulator [14] and the POX controller. The entire emulation process was executed on a single machine (Ubuntu 20.04.2 LTS 64-bit) equipped with a 12-core Intel(R) Core(TM) i7-9750H CPU running at 2.6 GHz and 16 GB of DDR4 RAM. The network topology is imported from the Fast Network Simulation Setup (FNSS) [26], and the FPLF algorithm is implemented using the POX controller as POX modules. To generate network traffic from servers within a DCN, the D-ITG [5] platform is employed. The overall architecture is illustrated in Figure 3.3.
To demonstrate the adaptive primary function of the FPLF algorithm in proportion to traffic demand, we conducted experiments with various scenarios of traffic demand, Low and high Traffic.

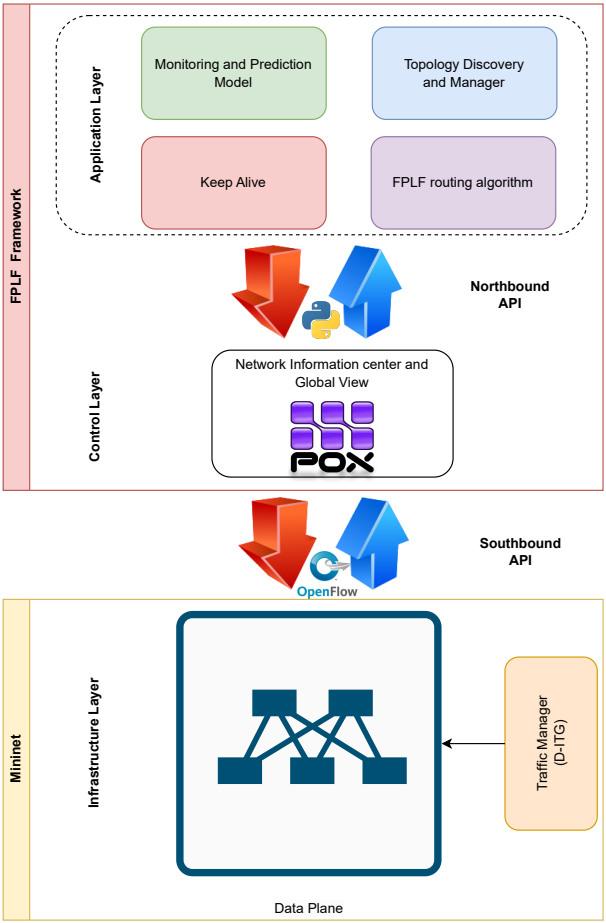

Figure 3.3: Architecture of the proposed framework and its components: the primary contribution is the FPLF algorithm and its traffic-aware optimizer blocks. OpenFlow was utilized on the southbound interface, and POX Python APIs were employed on the northbound interface.

3.3 Experiment Result

The result reported that the FPLF algorithm is suitable for deployment in DCNs that serve various traffic patterns, accommodating both low and high traffic loads. For instance, during periods of low traffic, such as night times, the FPLF algorithm demonstrates the potential to save up to 65.625% of the total energy consumed, leading to reduced network operating costs. On the other hand, during periods of high usage, such as rush hours, the FPLF algorithm efficiently distributes traffic load among switches with low latency, ensuring smooth transitions between traffic patterns and maintaining network performance.

3.3.1 QoS Criteria

This study considered two critical QoS criteria: (1) the algorithm response time, i.e., latency and (2) the number of dropped packets.

The FPLF algorithm exhibits a rapid response (i.e., low latency) to unexpected changes in traffic over time compare to ECMP (Equal Cost Multiple Plath), as shown in Figure 3.4, and 3.5. This attribute can be attributed to the robust monitoring system, reflecting the instantaneous network state, which enables the FPLF algorithm to take prompt actions that align with the traffic demand at specific moments.

On the other hand, because FPLF addresses power consumption by consolidating flows in a single link, this may affect the rate of packet arrivals at the destination. Therefore, a crucial metric describing the FPLF performance in *high traffic* scenarios is the number of dropped packets.

During the power consumption level testing, DITG decoder logs were used to record the results. The result showed that the FPLF outperformed the state-of-the-art algorithm in most cases.

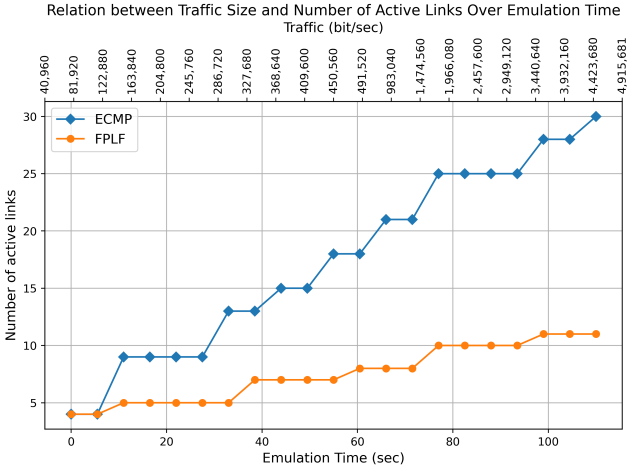

Figure 3.4: DCN link usage under low traffic load scenarios.

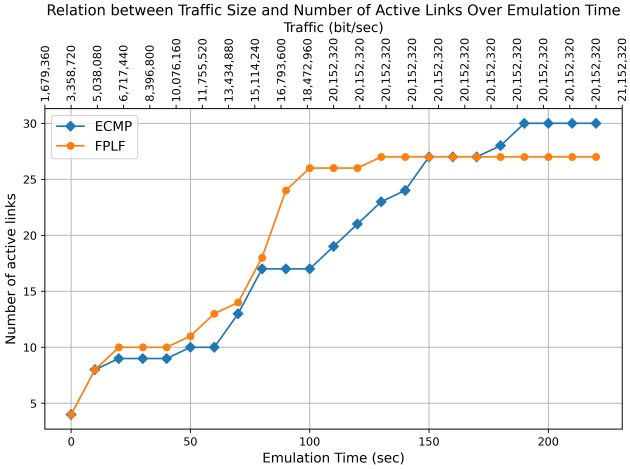

Figure 3.5: DCN link usage under high traffic load scenarios.

4 Thesis 3: Mitigation of the Effects of Consolidation Technique on the Quality of the RTAP Classes

In the preceding chapter, we introduced a novel routing strategy designed to enhance power efficiency in SD-DCNs. This strategy can blindly balance traffic flows between the core switches under high traffic conditions, without considering the specific services of the flow classes. However, certain classes, such as VoIP, video streaming, and Real-Time Protocol Applications (RTPA), can suffer from increased delays when mixed with other types of traffic. Consequently, we infer that such an algorithm could potentially impact the performance of real-time protocol applications. This realization prompted us to prioritize services for these specific classes (e.g., find an alternative path that meets their requirements), necessitating the development of a robust monitoring system capable of effectively identifying and rerouting such traffic to paths that meet their requirements. The central component of this monitoring system is the traffic classifier. Such a classifier plays a pivotal role in enhancing the performance of SD-DCNs. Therefore, this chapter proposes learning algorithms to classify a wide range of traffic in SDN environments. This work

represents a preliminary step towards optimizing QoS for flow classes within the framework of SD-DCNs.
This is accomplished by creating a dataset that encompasses various types of network traffic, including Video, Voice over IP (VoIP), Gaming, and Internet Control Message Protocol (ICMP). We evaluated the performance of several Machine Learning techniques, such as Linear Regression, Support Vector Machine, and Neural Network, and the results were presented in [19]. Moreover, we discussed the integration of classification into the power consumption model, aiming for further advancements in this research area.
It is worth noting that a substantial portion of the content in this chapter draws upon the insights presented in the work of [19].

4.1 D-ITG Dataset

To evaluate the proposed classification methods, we meticulously constructed a dataset using a synthetic traffic generator. Initially, we set up a small-scale network topology with the Mininet emulator [14], which included two hosts, one switch, and one central controller. Each host in the network was connected to an OpenFlow switch [18], and the OpenFlow switch was securely connected to a Ryu controller through an OpenFlow channel. As visualized in the Figure 4.1.
In the application layer, we developed two crucial models: (1) a monitoring model[1], designed to capture instantaneous flows, and (2) a collecting model, tasked with archiving traces for subsequent division into training and testing datasets.
The traffic flows were generated by the D-ITG generator tool [5]. D-ITG replicated the statistical properties of six traffic applications, including DNS, Telnet, Ping, Gaming, Quake-3, and VoIP. Additionally, we simulated video traffic using VLC to establish

[1]We utilized the same model discussed in Section 3.1.1. The distinction lies in solely collecting the aggregation matrix, **A**, without incorporating additional functions.

real-time video streaming between a VLC[2] server and client. As the traffic passed through the network switch, the monitoring model sent data to the collecting model, which extracted and archived the statistics of the arrival flows.
The primary function of the collecting model is to derive new features from the raw data received every second from the monitoring model, represented by aggregation traffic, **A**. This raw data encompasses the aggregation matrix of packets/bytes in both the forward and backward directions between the source and destination.
These raw data are then used to construct a flow object with the following features: Forward Packets, Forward Bytes, Delta forward Packets: the cumulative number of packets observed since the last forward flow was seen, Delta forward bytes: the cumulative number of bytes observed since the last forward flow was seen, Forward Instantaneous Packets per Second, Forward Instantaneous Bytes per Second, Forward Average Packets per Second, Forward Average Bytes per second, Reverse Packets, Reverse Bytes, Delta Reverse Packets, Delta Reverse Bytes, Reverse Instantaneous Packets per Second, Reverse Instantaneous Bytes per Second, Reverse Average Packets per second, and Reverse Average Bytes per second.
We ran each traffic application individually for approximately 150 minutes, resulting in up to 17,000 observations and 16 features for each experimental flow.

4.2 Classification Models

This section presents the implementation of three ML models, namely logistic regression, SVM, and NN with different types of solvers. We used Python 3.8 and Windows 10 with a 12-core Intel(R) Core(TM) i7-9750H CPU running at 2.6 GHz and 16 GB of DDR4 RAM to implement these models.

[2]VLC is a free and open source cross-platform multimedia player and framework that plays most multimedia files, and various streaming protocols.

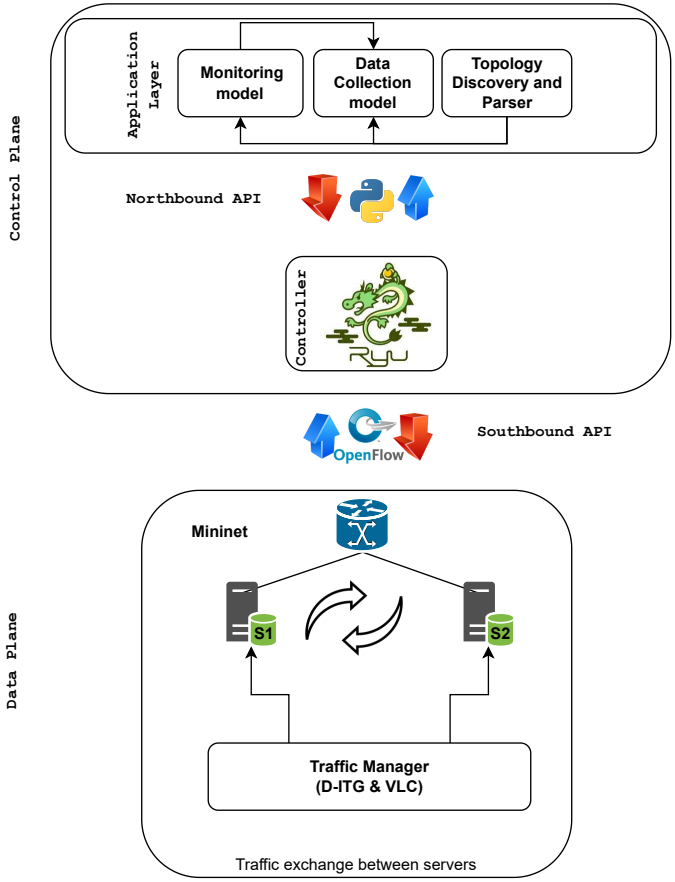

Figure 4.1: The structure of the SDN network in the data collection phase, including the monitoring and the collecting models. Python is used on the northbound interface, and OpenFlow is used on the southbound interface.

The test of the models is performed on a portion of the dataset (i.e., test data). It is worth mentioning here that we dropped the features of “Forward Byte," “Forward Packets," “Reverse Byte," and “Reverse Packet." This is due to the cumulative nature of those features. They may have significantly different values from those that occur in the training phase. For instance, while those features start counting from zero in the training phase, they may start from another initial value in real traffic, which depends on the dynamic behavior of the network. Therefore, ML models might be negatively affected by the misleading information of such features. We aim to ensure the model's robustness and generalizability to real-world scenarios.

4.2.1 Logistic Regression model

To construct a Logistic Regression model, the input is the dot product between a weight vector $\theta \in \mathbb{R}^n$ and an input vector $\mathbf{x}$ (i.e., input training data $x_k \in \mathbb{R}^n$) plus a bias b_k. And the output is the probability $\hat{y}_k \in \mathbb{R}$ to select the class k that maximizes (4.1), see [12]:

$$p\left(\mathbf{y}_k = 1 \mid \mathbf{x}\right) = \frac{\exp\left(\theta_k \cdot \mathbf{x} + b_k\right)}{\sum_{j=1}^{N} \exp\left(\theta_j \cdot \mathbf{x} + b_j\right)} \tag{4.1}$$

where, N is the number of classes.
For the purposes of regularization, feature selection, and providing better long-term predictions, we use Ridge Regression (i.e., ℓ_2 Regularization) and Lasso Regression (i.e., ℓ_1 Regularization) [3, 17]. The inverse of regularization strength (C) set default value is 1.0.
After training, we observed that models with the Stochastic Average Gradient Descent (SAGA) solver achieved accuracy up to 75%.

4.2.2 SVM Model

Like in LR model, we train the SVM model with input training data $x_k \in \mathbb{R}^n$ and output label data $y_k \in \mathbb{R}$ with ($N = 7$) classes

to construct the Support Vector Classifier (SVC) in the form [11]:

$$y(x) = \text{sign}\left[\sum_{k=1}^{N} \alpha_k y_k \psi\left(x, x_k\right) + b_k\right] \tag{4.2}$$

where α_k is a positive real constant, $\psi\left(x, x_k\right)$ represents the kernel, and b_k is the bias.
We trained the model with different types of kernels. The most noteworthy result from the data is that the Radial Basis Function (RBF) kernel achieved higher accuracy in less time compared to the other kernels. This finding highlights the efficiency of the RBF kernel in achieving superior accuracy up to 85% within a shorter time frame.

4.2.3 Neural Network model

In order to construct a NN model for the classification task, we stack 7 sequential layers as shown in Table 4.1.

Table 4.1: Description of the NN model.

Layer type	**Output Shape**	**Param #**
Normalization	(None, 12)	25
Dense	(None, 24)	312
Dense	(None, 48)	1200
Dropout	(None, 48)	0
Dense	(None, 28)	1372
Dense	(None, 14)	406
Dense	(None, 7)	105
Total params: 3420		
Trainable params: 3395		
Non-trainable params: 25		

The model consists of the following layers:

(1) Layer normalization [2] to scale and shift input values with zero mean and unit variance. The layer normalization

statistics are computed over all the input values $x_1, \ldots, x_n$ as follows:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

(2-3) Densely connected layers of size 24 and 48, respectively, using the Rectifier Linear Unit (ReLU) activation function $f(x) = \max(0, x)$.

(4) A dropout layer with 50% dropping rate, for the sake of regularization and to avoid overfitting.

(5-6) Densely connected layers of size 28 and 14, respectively, using ReLU.

(7) A densely connected layer of size 7, followed by a softmax activation (4.3), which outputs one of the $N = 7$ class labels.

$$\sigma(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{N}\exp(x_j)} \tag{4.3}$$

During training, we use the sparse categorical cross entropy loss function to cope with this multi-class classification problem where labels are provided as integers.
As an optimization technique for gradient descent, we ran experiments using both the Stochastic Gradient Descent (SGD) method and the Adaptive Moment Estimation (Adam) algorithm, achieving accuracies of up to 80% and 89%, respectively.

4.3 Online Testing

In this section, we delve into the evaluation of the NN model for predicting the labels of newly arriving flows in a real-time network scenario. To accomplish this, we seamlessly integrated the NN model with the SDN controller, alongside monitoring

and topology discovery, as depicted in the online workflow illustrated in Figure 4.2. Subsequently, we simulated each traffic type for approximately 10 minutes, initiating a burst of 1000 flows for each traffic type.
The testbed was configured using a Linux 20.04 LTS machine with 8 GB RAM and a Core i7 processor. We established the network topology using the Mininet emulator, with Ryu serving as the adopted network controller.

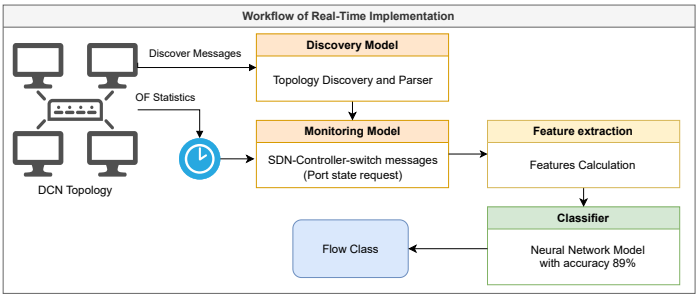

Figure 4.2: Workflow of the real-time implementation of the Neural Network classifier in an SDN environment.

Figure 5.1 illustrates the accuracy percentages for each traffic class. Notably, the classification model achieves 100% accuracy in predicting quake-3 traffic. Similarly, it excels in predicting DNS, video, and ping traffic classes, with accuracies reaching up to 99%, 97%, and 93%, respectively.
However, challenges arise in the classification of VoIP traffic, where the model fails to classify up to 14% of the injected traffic. Likewise, the model struggles with telnet traffic, misclassifying up to 35% of generated telnet traffic. The lowest performance is reported in predicting game traffic, with an accuracy of up to 13%.

5 Conclusion

In conclusion, our proposed suggestion model effectively reduces power consumption by selecting active subsets of the topology, with the next challenge being its real-world application. To address this, we introduce a heuristic algorithm aimed at minimizing active links in DCNs, enhancing efficiency through optimal path selection. Additionally, the FPLF algorithm balances power savings and performance in SDN-based DCNs using SDN controllers and the OpenFlow protocol. Lastly, to mitigate the impact of consolidation techniques in real-time applications, we explored machine learning models for traffic classification in SDNs, incorporating the NN model into an SDN implementation with promising accuracy. Future plans include enhancing FPLF's flow estimation model to improve performance, energy savings, and scalability, and combining the ML classification model with FPLF to optimize power consumption in data centers.

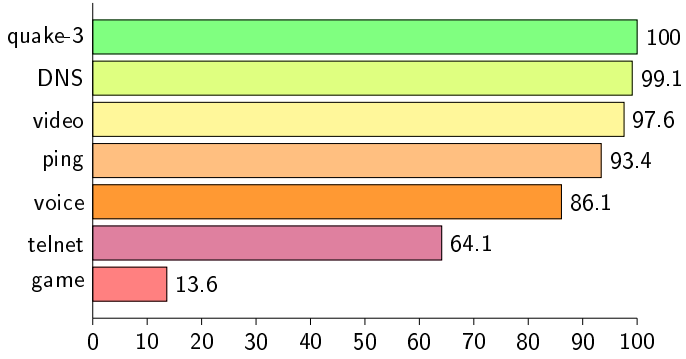

Figure 5.1: Result of SDN testing

Bibliography

[1] Beakal Gizachew Assefa and Öznur Özkasap. A survey of energy efficiency in sdn: Software-based methods and optimization models. *Journal of Network and Computer Applications*, 137:127–143, 2019. URL: https://doi.org/10.1016/j.jnca.2019.04.001.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL: https://doi.org/10.48550/arXiv.1607.06450.

[3] Richard G Baraniuk. Compressive sensing [lecture notes]. *IEEE signal processing magazine*, 24(4):118–121, 2007. URL: https://doi.org/10.1109/MSP.2007.4286571.

[4] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021. http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

[5] Alessio Botta, Walter de Donato, Alberto Dainotti, Stefano Avallone, and Antonio Pescapé. D-ITG 2.8. 1 Manual. *Computer for Interaction and Communications (COMICS) Group*, pages 3–6, 2013. URL:

https://www.traffic.comics.unina.it/software/ITG/manual/D-ITG-2.8.1-manual.pdf.

[6] Javier Arribas Cámara and Vicente Sánchez Jiménez. The european union facing the abyss: legislative review in the face of the energy crisis, 2022. *Journal of Energy & Natural Resources Law*, pages 1–16, 2023. URL: https://doi.org/10.1080/02646811.2023.2177409.

[7] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022. URL: https://doi.org/10.1145/3544585.3544600.

[8] John Forrest, Ted Ralphs, Haroldo Gambini Santos, Stefan Vigerske, Lou Hafer, John Forrest, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, rlougee, jpgoncal1, Jan-Willem, h-i gassmann, Samuel Brito, Cristina, Matthew Saltzman, tosttost, Fumiaki MATSUSHIMA, and to st. coin-or/cbc: Release releases/2.10.7, January 2022. URL: https://doi.org/10.5281/zenodo.5904374.

[9] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. https://www.gurobi.com/documentation/9.5/refman/, 2022.

[10] Ralph Hintemann and Simon Hinterholzer. Energy consumption of data centers worldwide. *Business, Computer Science (ICT4S)*, 2019. URL: https://www.researchgate.net/publication/341427004.

[11] Xiaolin Huang, Andreas Maier, Joachim Hornegger, and Johan AK Suykens. Indefinite kernels in least squares support vector machines and principal component analysis. *Applied and Computational Harmonic Analysis*, 43(1):162–172, 2017. URL: https://doi.org/10.1016/j.acha.2016.09.001.

[12] Ralf Klabunde. Daniel jurafsky/james h. martin, speech and language processing. *Zeitschrift für Sprachwissenschaft*, 21(1):134–135, 2002. URL: https://doi.org/10.1515/zfsw.2002.21.1.134.

[13] Gergely Kovásznai and Mohammed Nsaif. Integer programming based optimization of power consumption for data center networks. *Acta Cybernetica*, Nov. 2023. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/4335.

[14] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6, 2010. URL: https://doi.org/10.1145/1868447.1868466.

[15] Ali Malik and Ruairí de Fréin. A proactive-restoration technique for SDNs. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020. URL: https://doi.org/10.1109/ISCC50000.2020.9219598.

[16] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020. URL: https://doi.org/10.1126/science.aba3758.

[17] Gary C McDonald. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):93–100, 2009. URL: https://doi.org/10.1002/wics.14.

[18] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74, 2008. URL: https://doi.org/10.1145/1355734.1355746.

[19] Mohammed Nsaif, Gergely Kovásznai, Mohammed Abboosh, Ali Malik, and Ruairí de Fréin. ML-Based Online Traffic Classification for SDNs. In *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, pages 217–222. IEEE, 2022. URL: https://doi.org/10.1109/CITDS54976.2022.9914138.

[20] Mohammed Nsaif, Gergely Kovásznai, Malik Ali, and Ruairí de Fréin. Survey of routing techniques-based optimization of energy consumption in SD-DCN. *Infocommunications Journal*, (Special Issue on Applied Informatics):35–42, 2023. URL: https://doi.org/10.36244/ICJ.2023.5.6.

[21] Mohammed Nsaif, Gergely Kovásznai, Anett Rácz, Ali Malik, and Ruairí de Fréin. An adaptive routing framework for efficient power consumption in software-defined datacenter networks. *Electronics*, 10(23):3027, 2021. URL: https://doi.org/10.3390/electronics10233027.

[22] Rastin Pries, Michael Jarschel, Daniel Schlosser, Michael Klopf, and Phuoc Tran-Gia. Power consumption analysis of data center architectures. In *Proc. Int. Conf. on Green Communications and Networking (GreeNets)*, volume 51 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 114–124. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-33368-2_10.

[23] Furkan Rabee, A Al-Haboobi, and Mohammed Ridha Nsaif. Parallel three-way handshaking route in mobile crowd sensing (PT-MCS). *J Eng Appl Sci*, 14:3200–3209, 2019.

[24] Furkan Rabee, Mohammed Nsaif, and Ali Al-Haboobi. Reliable compression route protocol for mobile crowd sensing (RCR-MSC). *Journal of Communications*, 14:pp. 170–178, 02 2019. URL: https://doi.org/10.12720/jcm.14.3.170-178.

[25] Andre Rios, Alberto J Gonzalez, Jesus Alcober, Javier Ozon, and Kayhan Z Ghafoor. Conferencing services in P2P networks: Trends and challenges. *Future Internet Services and Service Architectures*, pages 139–160, 2022.

[26] Lorenzo Saino, Cosmin Cocora, and George Pavlou. A toolchain for simplifying network simulation setup. In *Proceedings of the 6th International ICST Conference on Simulation*

Tools and Techniques, SIMUTOOLS '13, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[27] Statista. Worldwide data created and consumed. https://www.statista.com/statistics/871513/worldwide-data-created/, 2023. 21-10-2023.

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

Registry number: DEENK/347/2024.PL
Subject: PhD Publication List

Candidate: Mohammed Nsaif
Doctoral School: Doctoral School of Informatics
MTMT ID: 10093904

List of publications related to the dissertation

Foreign language scientific articles in Hungarian journals (2)

1. Kovásznai, G., **Nsaif, M.**: Integer Programming Based Optimization of Power Consumption for Data Center Networks.
   Acta Cybern. Epub (-), 1-17, 2024. ISSN: 0324-721X.
   DOI: https://doi.org/10.14232/actacyb.299115
   IF: 0.4 (2022)

2. **Nsaif, M.**, Kovásznai, G., Malik, A., de Fréin, R.: Survey of Routing Techniques-Based Optimization of Energy Consumption in SD-DCN.
   Infocommun. J. 15 (Special), 35-42, 2023. ISSN: 2061-2079.
   DOI: http://dx.doi.org/10.36244/ICJ.2023.5.6
   IF: 1.1 (2022)

Foreign language scientific articles in international journals (2)

3. **Nsaif, M.**, Kovásznai, G., Malik, A., de Fréin, R.: SM-FPLF: Link-State Prediction For Software-Defined DCN Power Optimization.
   IEEE Access. Epub, 1-24, 2024. ISSN: 2169-3536.
   DOI: http://dx.doi.org/10.1109/ACCESS.2024.3408672
   IF: 3.9 (2022)

4. **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 (Switzerland). 10 (23), 1-18, 2021. EISSN: 2079-9292.
   DOI: http://dx.doi.org/10.3390/electronics10233027
   IF: 2.69

Foreign language conference proceedings (1)

5. **Nsaif, M.**, Kovásznai, G., Abboosh, M., Malik, A., de Fréin, R.: ML-Based Online Traffic Classification for SDNs.
   In: 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), IEEE, Piscataway, 217-222, 2022. ISBN: 9781665496537

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

List of other publications

Foreign language scientific articles in international journals (3)

6. **Nsaif, M.**, Mohammed, F. A., Abbas, F. M.: Detection and Prevention Algorithm of DDoS Attack Over the IOT Networks.
   TEM J. 9 (3), 899-906, 2020. ISSN: 2217-8309.
   DOI: http://dx.doi.org/10.18421/TEM93-09

7. Rabee, F., Al, H. A. S. A., **Nsaif, M.**: Parallel Three-Way Handshaking Route in Mobile Crowd Sensing (PT-MCS).
   J. Eng. Appl. Sci. 14 (10), 3200-3209, 2019. ISSN: 1816-949X.
   DOI: http://dx.doi.org/10.36478/jeasci.2019.3200.3209

8. **Nsaif, M.**, Al, H. A. S. A., Rabee, F., Alasadi, F. A.: Reliable Compression Route Protocol for Mobile Crowd Sensing (RCR-MSC).
   J. Communications. 14 (3), 170-178, 2019. ISSN: 2374-4367.
   DOI: http://dx.doi.org/10.12720/jcm.14.3.170-178

Total IF of journals (all publications): 8,09
Total IF of journals (publications related to the dissertation): 8,09

The Candidate's publication data submitted to the iDEa Tudóstér have been validated by DEENK on the basis of the Journal Citation Report (Impact Factor) database.

11 June, 2024