---

**Proceedings Paper:**
Alhindi, A. and Djemame, K. orcid.org/0000-0001-5811-5263 (Accepted: 2024) SDN Traffic Flows in a Serverless Environment: a Categorization of Energy Consumption. In: 17th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2024). 17th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2024), 16-19 Dec 2024, Sharjah, UAE. IEEE/ACM (In Press)

---

# SDN Traffic Flows in a Serverless Environment: a Categorization of Energy Consumption

Abdulaziz Alhindi
*School of Computing, University of Leeds, UK*
*Qassim University, KSA*
a.allhndi@qu.edu.sa

Karim Djemame
*School of Computing, University of Leeds, UK*
K.Djemame@leeds.ac.uk

***Abstract*— The widespread adoption of Software Defined Networks (SDN) in Information and Communication Technology (ICT) sectors has positioned it as a crucial technology due to its flexibility and modularity, which results from shifting control from physical network devices to software controllers. However, this shift has led to several challenges, including increasing energy consumption. Serverless computing can effectively support SDN to address various issues, such as energy efficiency, yet lacks an energy-aware scheduling framework designed for such an environment. This research investigates the key factors influencing serverless energy consumption in SDN by examining the correlations between the energy usage of serverless applications and traffic flow patterns, as such correlations can play a critical role in developing an energy-aware scheduling framework. The findings reveal that several traffic patterns strongly or moderately impact energy consumption in serverless environment.**

**Keywords—Serverless, Energy Efficiency, Software Defined Network, ONOS, Knative, Scheduling, Flow Management.**

## I. INTRODUCTION

The rapid proliferation of smart devices worldwide has resulted in an unprecedented surge in data traffic, underscoring the challenges associated with controlling and managing information flow within networks [1,2]. Traditionally, networking infrastructures were primarily designed to handle text-based content and are therefore ill-equipped to manage the dynamic traffic and interactive applications characteristic of modern networks. To address the challenges posed by the explosive increase and heterogeneity in traffic, automatic and adaptive network management strategies can be employed. These strategies improve scalability, reliability, and cost-effectiveness while maintaining the required quality of service [3,4].

Software Defined Networks (SDN) has emerged as a promising technology capable of providing effective solutions for managing the vast amounts of heterogeneous data traversing contemporary networks [5]. By decoupling the control logic from the forwarding devices, SDN enables more efficient resource management and establishes a flexible network management framework. In this framework, remote software can control and program forwarding devices according to network policies set by administrators. SDN simplifies network management by abstracting away the underlying complexities of network infrastructure, thereby allowing it to meet the increasing demands of networking applications. Key advantages of SDN include ease of routing management and flexibility in implementing network policies.

While SDN offers several solutions to the challenges of traditional networks, it also encounters several difficulties, which have become active areas of research in recent years. One significant issue is the rising energy demands resulting from the increased use of software controllers and servers in data centres [6]. Research has shown that energy costs account for over 10% of the operating expenses for ICT service providers.

Moreover, the widespread adoption of cloud services and cloud-based technologies, such as containers and microservices, has facilitated the rise of a new programming paradigm known as serverless computing, or Function as a Service (FaaS) [7]. This model leverages container-based virtualization to provide a platform that allows software developers to deploy applications with minimal management of the underlying infrastructure. A key feature of serverless platforms is their high level of abstraction, wherein applications are broken down into fine-grained functions that are deployed and executed without developer intervention. Unlike traditional cloud services, serverless functions are executed only when invoked, which provides a resource-efficient, low overhead alternative to Virtual Machines (VMs) and containers [8,23]. Consequently, serverless computing is seen as a promising approach for enhancing the efficiency of cloud/edge resource utilisation, reducing energy consumption, and lowering the cost of resource allocation compared to VMs and containers [23].

This paper forms the first part of ongoing research aimed at enhancing the energy efficiency of SDN by utilising serverless computing to manage and execute non-core functions of SDN controllers. The research aims to develop an energy-aware scheduling framework for serverless applications in SDN, leveraging machine learning models to optimize scheduling decisions. These models will be developed based on the primary factors that impact the energy consumption of serverless applications, allowing for more accurate scheduling decisions that can significantly reduce energy usage. Given that SDN applications primarily handle network traffic, this paper examines the relationships between traffic patterns and the energy consumption of serverless applications in SDN environments.

The primary objective of this paper is to propose a method for identifying key correlations between the energy consumption of a serverless application developed for network-related tasks and the characteristics of the incoming traffic handled by this application. Understanding these correlations is crucial for improving the energy efficiency of serverless applications within SDN environments, as they offer valuable insights into the application's energy consumption. These insights can potentially guide decisions regarding optimal placement or resource consumption prediction for serverless functions. This paper therefore presents the methodology employed to discover these correlations and highlights the results, which identify the most influential traffic features affecting the application's energy consumption.

The contributions of this paper are:

1. A comprehensive presentation of the methodology used to identify the traffic features that most significantly impact the energy consumption of serverless applications in SDN.
2. A quantitative analysis of the results, showcasing the most impactful traffic features.
3. A set of recommendations for adopting serverless computing in SDN in an energy-aware environment.

The remainder of this paper is organized as follows. Section 2 reviews some of the work on SDN, energy efficiency and serverless computing. Section 3 describes the methodology employed to investigate the correlations between the energy consumption of serverless applications in SDN, and traffic characterisation. The experimental design is presented in Section 4 and the results of the investigation are discussed in Section 5. Section 6 concludes the paper.

## II. Related Work

The energy efficiency of SDN has been a prominent research topic in recent years, with numerous studies exploring the issue from various perspectives. Research in this field typically focuses on four main areas: traffic awareness, end-host awareness, rule placement, and controller placement.

Reference [18] addresses an aspect of the controller placement problem, which often involves dividing the network into several sub-networks, depending on its size, and assigning each partition to a dedicated controller for management. To reduce energy consumption, the study proposes determining the optimal number of network partitions that can meet the required latency while avoiding underutilisation of controllers and links. The proposed solution connects controllers with switches through in-band control planes, differing from the traditional out-of-band control planes that establish static links between controllers and switches. This approach is specifically designed to handle dynamic and large volumes of data traffic generated by Internet of Things (IoT) devices. The evaluation of this solution demonstrated an energy saving of approximately 22%.

Reference [19] explores enhancing the energy efficiency of SDN controllers by utilising multi-core processors instead of single-core ones. According to this study, energy consumption in SDN is distributed across three components: controllers, switches, and links. While switches and links account for the majority of energy consumption, optimizing controller energy use can also enhance the overall energy efficiency of the network. The conducted experiment shows that processing a workload using multiple cores operating at a lower frequency consumes less energy than using a single core at a higher frequency. As a result, an end-host-aware solution is proposed that implements a parallelised controller to distribute the load across multiple processor cores while lowering the frequency. This trade-off between performance and energy consumption resulted in a 28% reduction in energy consumption.

Another study examines an energy-efficient approach for balancing the load across SDN controllers [20]. The traffic-aware solution migrates some of the load from heavily burdened controllers to lightly loaded ones, which includes transferring some switches as well. This migration process requires an energy-efficient re-routing algorithm to reduce energy consumption while maintaining service levels. The developed system comprises several components that continuously monitor the load of each controller and ensure that every controller is aware of the others' loads. This enables each controller to calculate and update the threshold at which the load migration process is triggered. Their experimental results show that the proposed solution achieved a 25% reduction in energy consumption along with a 20% improvement in performance.

Several studies have demonstrated that serverless computing is both more energy-efficient and more effective in managing resources compared to other platforms, such as Docker [13,14]. Consequently, various researchers have proposed serverless computing as a solution for managing and deploying SDN applications. For example, the work of [15] introduced a new SDN architecture that incorporates serverless computing to process most of the events originating from the data plane. This solution builds on the architecture proposed in [16], where the SDN controller is reduced to perform only essential tasks, while other functions are offloaded to separate modules executed by the serverless framework.

This study distinguishes itself from previous research by examining the correlations between energy consumption of serverless applications in SDN and the associated traffic patterns. The significance of these correlations lies in their potential to predict the energy consumption of the serverless application, which can be leveraged to inform the development of an energy-aware scheduling framework for serverless applications in SDN.

## III. Methodology

### A. Architecture

SDN is designed to separate the control plane from the data plane, promoting scalability and simplifying network management [15]. The control plane manages core functionalities and applications like firewalls and load balancing through Northbound APIs, while the data plane includes forwarding devices, using OpenFlow as a Southbound API. Traditional SDN controllers are monolithic, making it difficult to add services independently. To address this, a modular SDN architecture using serverless computing can be used to allow external applications to provide services as independent, scalable functions, enabling flexible resource management and network programmability (see Fig. 1).

To fully integrate serverless computing into SDN, a bridging mechanism can be employed to connect the SDN controller with the serverless platform. This bridge is responsible for capturing the required packets and distributing them to the appropriate serverless applications. Additionally, the bridge facilitates the transfer of actions generated by the serverless applications back to the controller, which then applies these actions to the data plane devices. This approach ensures seamless communication and control between the SDN controller and serverless platform. The source code of this bridge can be found on the project's GitHub page [24].

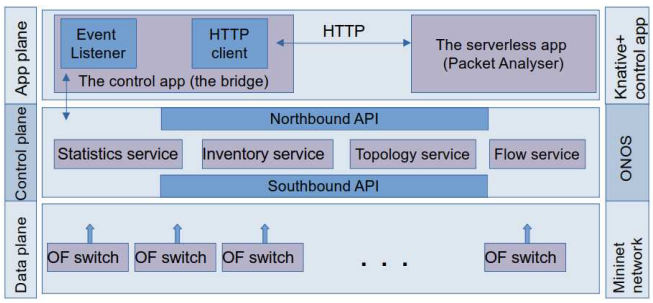


Fig. 1. SDN architecture in a serverless environment

### B. Design Method

As the primary objective of this paper is to investigate the key correlations between the energy consumption of serverless applications in SDN and the characteristics of their incoming traffic, this section outlines the methodology employed to achieve this goal.

Network traffic consists of multiple flows, each representing the packets exchanged between two IP addresses over a specific period. Thus, mapping energy consumption to data traffic involves answering the question: how to measure the energy consumed by the serverless application to process each flow?

The serverless application concurrently processes varying numbers of flows over time. Additionally, traffic flows vary significantly in duration, with some lasting only a few microseconds, while others may persist for several seconds or longer. This variability provides a pathway for mapping the energy consumed in processing traffic flows to individual flows, acknowledging that the serverless application may simultaneously process packets from multiple flows. Energy consumption, measured in Joules, is the product of power usage over time, where one Joule equals one watt-second.

We propose a method to measure the energy consumed by the serverless application to process a single traffic flow. The first step is to define a time interval during which the energy consumption of the serverless application is measured, along with the number of processed packets and their corresponding flows. The next step involves dividing traffic flows based on the selected time interval. Some flows may start and end within a single time interval (discontinuous flows), while others may span multiple time intervals (continuous flows), as illustrated in Fig. 2. The energy consumed to process discontinuous flows can be measured within a single time interval, while the energy consumed by continuous flows must be aggregated over several time intervals.

The final step is to allocate the energy consumed during each time interval across the processed traffic flows based on their relative weights. These flows can be either continuous, where only a portion is processed within the time interval, or discontinuous, where the entire flow is processed within the time interval. Each flow's weight within a given time interval is calculated by multiplying its duration by its speed (packets per second), and then dividing the result by the total duration of all flows in that interval. The duration of a discontinuous flow is the time difference between its start and end, while the duration of a continuous flow is the length of the flow segment within the time interval. Ultimately, each discontinuous flow will have an estimated energy consumption for its processing, while the energy consumption of continuous flows will be the sum of the energy consumed by its segments processed across multiple time intervals.

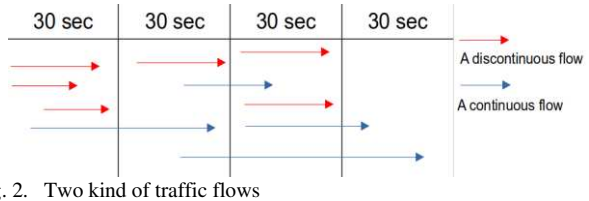


Fig. 2. Two kind of traffic flows

### C. Hypotheses

Throughout the experimental design, the following hypotheses were formulated based on observation of similar trends in the literature:

1. **H1**: The number of processed packets is one of the most significant factors influencing the energy consumption of the serverless application.

2. **H2**: The length of the processed packets has a strong correlation with the energy consumption, as large packets require more energy to be processed.

3. **H3**: The rate at which packets are transmitted has a significant correlation with the energy consumption, as high transmission speeds may increase the workload on the serverless application, leading to greater energy usage.

4. **H4**: The energy consumption of serverless applications can be accurately predicted based on specific traffic patterns or features.

### D. Software Tools

The technologies used in the experimental environment setup for SDN management control and the serverless computing platform are described next.

#### 1) Knative

Knative [11] is an open-source serverless framework initially developed by Google in collaboration with over 50 technology companies, and it is currently hosted by the Cloud Native Computing Foundation. The framework is primarily designed as an extension to Kubernetes, providing a platform for building and managing containerised serverless applications. These applications can be deployed in two main forms: Serving, which consists of a set of serverless functions along with their associated resources, such as configurations and revisions, and Eventing, which is more complex and facilitates the deployment of an event-driven framework for managing the exchange of events between serverless functions.

As this research focuses on testing a single serverless application for processing events received from ONOS, Knative Serving—whose architecture is illustrated in Fig. 3—was utilised for deploying the application.

#### 2) ONOS

The Open Network Operating System (ONOS) [12] is a leading open-source controller designed for managing SDN and Network Functions Virtualization (NFV) environments. ONOS is built to support distributed deployment, ensuring scalability, high availability, and simplified network management. Additionally, it provides a modular framework that allows developers to create extensions for performing various network tasks with ease. The strong community support and the platform's flexibility in enabling modular network management are the primary reasons this controller was selected for use in this research.

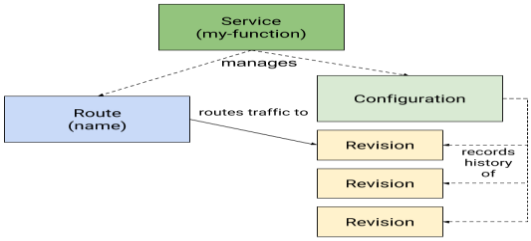


Fig. 3. Knative Service Architecture [11]

## IV. Experimental Design

The implementation of the proposed methodology and underlying experiments that were conducted are described next.

### A. Experiment Description

The conducted experiment is comprised of five main components, all of which were deployed on a single machine, as shown in Fig. 4. This machine is equipped with the following hardware specifications: AMD CPU Ryzen 7 5800h with 8 cores and base clock 3.2GHz, 16 GB memory, Ubuntu 22.04 LTS OS, and 512 GB SSD storage.

*1) Serverless application:* Various network applications could have been implemented for this experiment, such as a firewall or an intrusion detection system. However, a packet analyser was chosen as the serverless application for this experiment. The application operates by inspecting the IP addresses and ports of both the source and destination and comparing them against a predefined list of blocked IPs and ports. Additionally, the application scans packet payloads using regular expressions to detect malicious patterns related to attacks like SQL injection. Each packet is then classified as either allowed or denied based on the results of these checks. This application is implemented as a single serverless function written in Python.

*2) Energy consumption measurement:* Several tools can be used to measure the energy consumption of serverless applications. For this experiment, Kepler [21] was chosen due to its integration with Kubernetes clusters, allowing it to track the energy consumption of individual pods or namespaces. Kepler collects the necessary metrics using Prometheus and presents the energy consumption data via Grafana.

*3) SDN controller:* for managing SDN, ONOS is chosen for this experiment.

*4) A virtual network:* A virtual network is setup whose primary purpose is to replay pre-captured traffic. Mininet was used to create and manage the network, consisting of two virtual hosts connected to a single virtual switch, with ONOS serving as the main controller.

*5) The control application*: It was necessary to develop a communication bridge or channel capable of exchanging events between ONOS and serverless applications, see Fig. 1. This bridge facilitates event exchange through the HTTP protocol, as serverless applications typically receive requests via HTTP. Consequently, a control application on ONOS that collects the required packets and forwards them to the serverless application was developed. Simultaneously, the control application receives responses from the serverless application and applies the necessary changes to the switches' rules.

### B. Pre-captured Traffic

It was crucial to use pre-captured traffic generated from SDN environments rather than traditional networks. The investigation of available datasets revealed that there are only a limited number of studies providing raw traffic datasets captured from SDN environments. One such dataset is presented in [17], where the authors emphasize the need for updated and compatible traffic datasets for SDN networks. This dataset, created by a team at University College Dublin in 2020, is designed for evaluating Intrusion Detection Systems (IDS) in SDN. It includes two types of SDN traffic: normal and attack traffic, with the attack traffic mimicking real-world scenarios from external and internal networks.

For this experiment, the *normal traffic dataset* was used. It contains 11 raw traffic files (pcap files) with over 50,000 flows and more than one million packets. The packet lengths in this dataset occasionally exceeded 64,000 bytes, which caused issues when processed by ONOS and the serverless application. Therefore, packets exceeding 16,384 bytes were filtered out using Tcpreplay [22], resulting in a reduction of approximately 2% of the flows in each pcap file—a negligible effect on the overall results. The dataset, then, was cleaned by removing flows with unreasonable values, such as negative flow durations. Once this process was completed, the number of flows was reduced to 17,946, of which 17,617 are TCP flows, while the remaining are UDP. The key metrics associated with these flows are presented in Table 1.

Table I. PRIMARY METRICS FOR EACH FLOW

| | **Metrics For Each Flow** | | | | | |
|---|---|---|---|---|---|---|
| | **No. of Packets** | **Total Packet Lengths (bytes)** | **Duration (seconds)** | **IAT Mean (seconds)** | **Packets /second** | **Packet Length Mean (bytes)** |
| **Mean** | 96.05 | 136,575.95 | 44.61 | 1.65 | 376.75 | 475.52 |

To extract traffic features, CICFlowMeter [10], developed by the Canadian Institute for Cybersecurity at the University of New Brunswick, was used. This tool extracts over 80 features per flow, including factors such as forward and backward packets, inter-packet timings, packet lengths, and more. The features were exported as CSV files and used as input to determine correlations with energy consumption.

### C. Energy Consumption Measurement

For this experiment, Kepler was chosen for its ability to estimate the energy consumption specific to the serverless application deployed on Kubernetes. It provides energy consumption metrics in intervals as short as 30 seconds, which served as the time periods within which energy consumption was divided across the observed flows.

### D. Experiment's Scenario

The goal of this experiment was to measure the energy consumed by the serverless application while processing traffic flows. The experimental scenario involved replaying pre-captured traffic within an SDN network while monitoring the energy consumption of the serverless application, as illustrated in Fig. 4. Tcpreplay was used to replay the traffic on one of the hosts, allowing the ONOS control application to capture each packet and forward it to the serverless application. While the serverless application processed the packets, Kepler recorded the corresponding energy consumption. This data was then used to create a dataset, with traffic features as independent variables and energy consumption as the dependent variable.

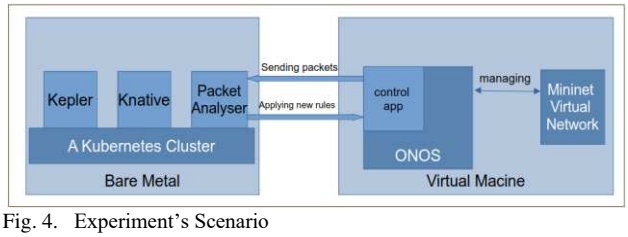


Fig. 4. Experiment's Scenario

A Python application was developed to identify the traffic features most correlated with energy consumption, using the Spearman correlation coefficient. The Spearman method was chosen due to its robustness against outliers, which are common in network flow data, and its ability to detect monotonic relationships potentially caused by these outliers

[9]. Also, outliers in this dataset were not disregarded since they often represent spikes in flow activity.

## V. Results and Discussion

The primary goal of this experiment was to identify the key factors influencing the energy consumption of the serverless application. The analysis revealed one feature with a correlation value exceeding 0.7, indicating positive strong correlations. Also, it showed other features with values either between 0.4 and 0.7, indicating positive moderate correlations, or between -0.4 and -0.7, which indicate negative moderate correlations [9].

The feature strongly correlated with energy consumption is the total number of packets in a flow, with a correlation value of 0.84. The following features exhibit moderate correlations:

1. Total sum of packet lengths (0.65)
2. Minimum inter-arrival time (IAT) between packets (-0.49)
3. Maximum packet length (0.48)
4. Flow duration (0.44)
5. Mean packet length (0.42)
6. Average payload size (0.41)
7. Standard deviation and variance of packet lengths (0.40).

Fig. 5 displays the above features, along with others that exhibit weak correlations with energy consumption. Details on the features can be found on the GitHub page of the CICFlowMeter app [10].

### A. Findings

It is expected that the total number of packets in a flow is the most impactful feature, as packets represent the primary input for the serverless application. Therefore, the energy consumed in processing a flow is expected to have a linear relationship with the number of packets in that flow (see Fig. 6), *which provides a confirmation of hypothesis H1*.

Additionally, the energy consumption of the serverless application increases when processing flows with longer packet lengths, as indicated by features such as maximum packet length, mean packet length, variance, standard deviation, and payload size. These features, with moderate correlations between 0.4 and 0.48, suggest that packet length and payload size are important factors for predicting energy consumption, *which provides a confirmation of hypothesis H2*. Predicting features such as the mean, variance, and standard deviation may simplify the prediction of total packet lengths, as the mean is highly correlated with total length (correlation value of 0.92, as shown in Fig. 7).

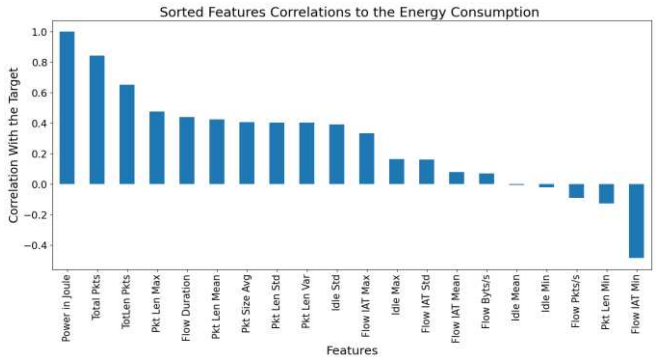


Fig. 5. The most correlated traffic features to energy consumption

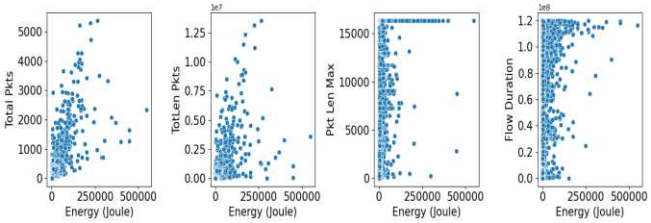


Fig. 6. Four correlated traffic features to energy consumption

On the other hand, the minimum IAT showed a negative correlation with energy consumption, indicating that shorter IATs result in more frequent packets, potentially requiring the serverless application to scale up replicas, which increases energy consumption. Flow duration had the weakest correlation among the examined features but is closely related to flow speed and IAT (Fig. 8), suggesting it could be predicted indirectly.

Contrary to the expectations of the third hypothesis, flow speed—whether measured in packets per second or flows per second—does not directly impact the energy consumption of the serverless application. However, the minimum IAT demonstrates a moderate correlation with energy consumption and may provide a more accurate representation of flow speed than packets per second. This is because CICFlowMeter calculates the number of packets per second by simply dividing the total number of packets by the flow duration, a basic calculation that does not fully capture the actual flow speed. *Hypothesis H3 is therefore partly confirmed.*

To demonstrate the strong correlation between energy consumption and the number of packets in a flow, Fig. 9 compares predicted energy consumption using linear/polynomial regression and Random Forest. The linear regression model performed well, especially for smaller energy values, as the predicted values closely aligned with the actual data (represented by the red line), *which provides a confirmation of hypothesis H4*.

By identifying flows that require more energy, serverless applications associated with processing high-energy flows can be scheduled to be executed on more powerful machines, while those processing low-energy flows can be scheduled on less energy-intensive machines. Furthermore, profiling flows based on energy consumption can play an important in predicting resource usage, which, in turn, can significantly reduce the energy consumption of serverless applications and help in selecting the optimal placement [25]. This approach can increase server utilisation, improving energy efficiency and optimising performance. Additionally, predicting the energy consumption of flows allows for proactive scaling of serverless function replicas before high-energy flows arrive, thereby mitigating the performance impact of cold starts.

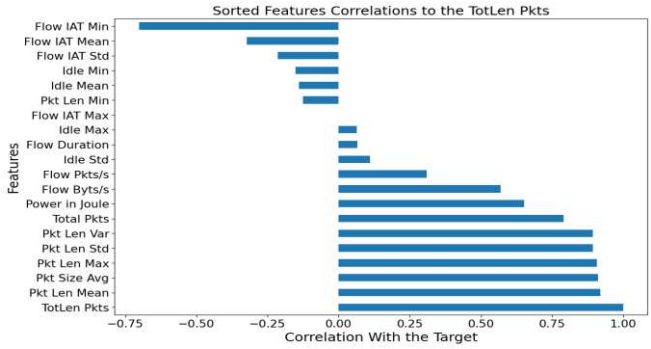


Fig. 7. The most correlated traffic features to total packet lengths

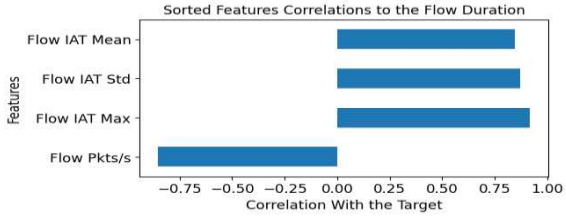


Fig. 8. The most correlated traffic features to flow duration.

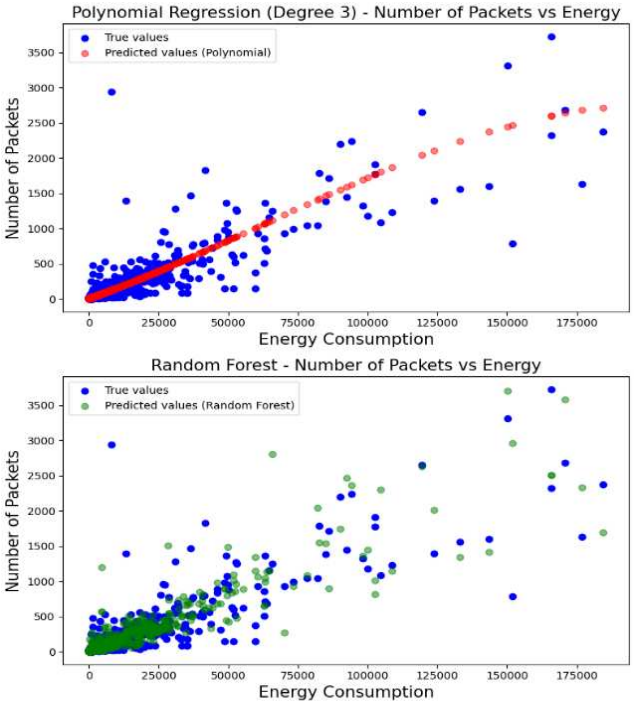


Fig. 9. Predicting energy consumption using the number of flow packets

## VI. Conclusion and Future Work

This paper explores the correlation between energy consumption of serverless applications in SDN environments and the characteristics of incoming traffic. By offloading non-core controller functions to serverless applications, the study aims to enhance the energy efficiency of SDN. An experiment was conducted using ONOS as the SDN controller and a serverless packet analyser, with energy consumption measured via Kepler. Pre-captured SDN traffic was replayed in a Mininet environment, and flow features were extracted for analysis. The results show that the number of packets in a flow is the most significant factor affecting energy consumption, with other factors such as packet length and IAT also having moderate correlations. These findings suggest that traffic management in SDN can be optimized to improve energy efficiency by predicting and managing flows based on their energy demands.

Although this paper provides valuable insights, further research is necessary to fully understand the energy consumption of flows. Identifying the most influential factors on SDN application's energy consumption, whether deployed on serverless platforms or others, can lead to more accurate energy consumption predictions in the future. Moreover, this research is an important milestone to design an energy-aware scheduling framework that leverages machine learning models to optimize scheduling decisions.

## References


[1] Al-Makhlafi, Moeen, et al. "RibsNet: A scalable, high-performance, and cost-effective two-layer-based cloud data center network architecture." IEEE Transactions on Network and Service Management 20.2 (2022): 1676-1690.

[2] Bannour, Fetia, Sami Souihi, and Abdelhamid Mellouk. "Distributed SDN control: Survey, taxonomy, and challenges." IEEE Communications Surveys & Tutorials 20.1 (2017): 333-354.

[3] Balakrishnan, Hari, et al. "Revitalizing the public internet by making it extensible." ACM SIGCOMM Computer Communication Review 51.2 (2021): 18-24.

[4] McCauley, James, et al. "Enabling a permanent revolution in internet architecture." Proceedings of the ACM Special Interest Group on Data Communication. 2019. 1-14.

[5] Singh, Sanjeev, and Rakesh Kumar Jha. "A survey on software defined networking: Architecture for next generation network." Journal of Network and Systems Management 25 (2017): 321-374.

[6] Assefa, Beakal Gizachew, and Öznur Özkasap. "A survey of energy efficiency in SDN: Software-based methods and optimization models." Journal of Network and Computer Applications 137 (2019): 127-143.

[7] Kritikos, Kyriakos, and Paweł Skrzypek. "A review of serverless frameworks." 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). IEEE, 2018.

[8] Fox, Geoffrey C., et al. "Status of serverless computing and function-as-a-service (faas) in industry and research." arXiv preprint arXiv:1708.08028 (2017).

[9] Rebekić, Andrijana, et al. "Pearson's or Spearman's correlation coefficient-which one to use?." Poljoprivreda 21.2 (2015): 47-54.

[10] CanadianInstituteForCybersecurity. "Cicflowmeter." GitHub, github.com/CanadianInstituteForCybersecurity/CICFlowMeter/blob/master/ReadMe.txt. Accessed 19 Sept. 2024.

[11] Knative, knative.dev/docs/. Accessed 19 Sept. 2024.

[12] "Open Network Operating System (ONOS)." Open Networking Foundation, 16 July 2024, opennetworking.org/onos/. Accessed 19 Sept. 2024.

[13] Alhindi, Abdulaziz, Karim Djemame, and Fatemeh Banaie Heravan. "On the power consumption of serverless functions: an evaluation of openFaaS." 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC). IEEE, 2022.

[14] Jia, Xuechao, and Laiping Zhao. "RAEF: Energy-efficient resource allocation through energy fungibility in serverless." 2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2021.

[15] Banaie, Fatemeh, and Karim Djemame. "A serverless computing platform for software defined networks." International Conference on the Economics of Grids, Clouds, Systems, and Services. Cham: Springer Nature Switzerland, 2022.

[16] Comer, Douglas, and Adib Rastegarnia. "Toward disaggregating the SDN control plane." IEEE Communications Magazine 57.10 (2019): 70-75.

[17] Elsayed, Mahmoud Said, Nhien-An Le-Khac, and Anca D. Jurcut. "InSDN: A novel SDN intrusion dataset." IEEE access 8 (2020): 165263-165284.

[18] Maity, Ilora, Ravi Dhiman, and Sudip Misra. "Enplace: Energy-aware network partitioning for controller placement in sdn." IEEE Transactions on Green Communications and Networking 7.1 (2022): 183-193.

[19] Oliveira, Tadeu F., Samuel Xavier-de-Souza, and Luiz F. Silveira. "Improving energy efficiency on SDN control-plane using multi-core controllers." Energies 14.11 (2021): 3161.

[20] Priyadarsini, Madhukrishna, et al. "An energy-efficient load distribution framework for SDN controllers." Computing 102.9 (2020): 2073-2098.

[21] "Kubernetes Efficient Power Level Exporter (Kepler)." Kepler, sustainable-computing.io/. Accessed 19 Sept. 2024.

[22] "TCPReplay." Tcpreplay, tcpreplay.appneta.com/. Accessed 19 Sept. 2024.

[23] Djemame, Karim. "Energy efficiency in edge environments: a serverless computing approach." Economics of Grids, Clouds, Systems, and Services: 18th International Conference, GECON 2021, Virtual Event, September 21–23, 2021, Proceedings 18. Springer International Publishing, LNCS 13072.

[24] "OnosBridge." OnosBridge, https://github.com/aalhindi/OnosBridge . Accessed 9 Nov. 2024.

[25] Stojkovic, Jovan, et al. "EcoFaaS: Rethinking the Design of Serverless Environments for Energy Efficiency." Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA'24). 2024.