

Flow-Aware, Machine Learning-Driven Mapping for Serverless Software Defined Networks

Abdulaziz Alhindi^{1,2} and Karim Djemame¹

¹*School of Computer Science, University of Leeds, U.K.*

²*College of Computing, Qassim University, K.S.A.*

Keywords: Software-Defined Networking, Serverless Computing, Machine Learning, Flow-Aware Mapping.

Abstract: Energy efficiency has become a critical challenge in Software-Defined Networking (SDN), particularly as modern controllers increasingly rely on computationally intensive software components. In parallel, serverless computing has appeared as a lightweight, event-driven execution model that offers fine-grained resource utilisation and reduced operational overhead. While recent studies have explored integrating serverless platforms into SDN architectures, they largely focus on architectural disaggregation and overlook energy-aware resource provisioning decisions. This paper proposes a Machine Learning (ML)-driven, flow-aware architecture for energy-aware resource provisioning in serverless SDN environments. The proposed approach predicts per-flow energy consumption, processing time, and CPU usage using machine learning models trained on flow-level features. These predictions are used to dynamically map incoming data flows to the most energy-efficient execution nodes. To support accurate prediction, we introduce a methodology for fine-grained measurement of per-flow energy consumption and processing time in serverless SDN environments. The architecture is evaluated using an SDN-serverless testbed built with ONOS and Knative, leveraging real SDN traffic traces. Experimental results demonstrate that the proposed solution effectively reduces energy consumption compared to baseline deployments while maintaining service quality, highlighting the benefits of flow-aware, ML-based resource provisioning in serverless SDN systems.

1 INTRODUCTION

Despite the improvements introduced by Software-Defined Networking (SDN) to traditional networks, SDN also introduces several challenges, a major one of which involves the rising energy consumption driven by the increasing reliance on software controllers and servers in data centres (Assefa and Özkasap, 2019). In parallel, serverless computing is increasingly regarded as a promising approach for improving the efficiency of cloud and edge resource utilisation, reducing energy consumption, and lowering the overall cost of resource allocation when compared with Virtual Machine (VM)- or container-centric models (Djemame, 2021; Alhindi et al., 2022; Djemame, 2022).

Accordingly, a growing body of recent research has explored architectural models for integrating serverless into SDN where non-essential controller functions are migrated from the control plane to the application plane where they are deployed and managed as serverless applications (Banaie et al., 2025;

Banaie and Djemame, 2022; Aditya et al., 2019). Despite these recent contributions, the existing body of research does not sufficiently examine how serverless computing can be practically deployed within SDN environments, nor does it identify which integration architectures can most effectively maximise the energy-efficiency advantages inherent to serverless systems.

Therefore, the main objective of this study is to introduce a novel architecture that integrates serverless computing into SDN while enabling energy-aware mapping of incoming data flows to their corresponding serverless applications and execution nodes. Rather than scheduling serverless functions, the proposed approach focuses on predicting the energy consumption and processing time of each flow and using these predictions to determine the most suitable node on which the target serverless application should process incoming data flows. To support these decisions, the proposed solution begins by introducing a methodology for collecting detailed measurements of energy and time needed by serverless functions when

processing data flows. These measurements are subsequently used to train a set of Machine Learning (ML) models capable of predicting energy consumption, processing time, and other relevant performance metrics for each incoming flow. The resulting predictions are then utilised by another component of the architecture to determine the most energy-efficient node on which each flow should be processed, ensuring minimal energy use without affecting performance.

The key contributions of this paper are as follows:

- A methodology for accurately measuring per-flow energy consumption, enabling fine-grained analysis of the energy required to process individual data flows in serverless environments.
- A novel energy-aware architecture for integrating serverless computing into SDN, in which each incoming data flow is assigned to an execution node based on an energy-aware mapping algorithm designed to minimise energy usage while preserving service constraints.
- A comprehensive evaluation of the proposed architecture, demonstrating the extent to which energy consumption can be reduced and validating the effectiveness of the energy-aware scheduling decisions.

2 RELATED WORK

Energy efficiency in SDN has attracted significant research attention in recent years, with numerous studies investigating the problem from multiple perspectives.

Reference (Maity et al., 2022) examines the controller placement problem by partitioning a network into sub-networks, each managed by a dedicated controller. The authors determine the optimal number of partitions that meet latency constraints while avoiding controller and link underutilisation, achieving about 22% energy reduction in experiments. In (Oliveira et al., 2021), the authors investigate improving the energy efficiency of SDN controllers by utilising multi-core processors instead of single-core architectures. Their results show that distributing workloads across multiple low-frequency cores reduces energy consumption, and the proposed parallelised controller approach achieves about 28% energy savings. Another study presents an energy-efficient load-balancing mechanism for SDN controllers where workload can be migrated from heavily loaded controllers to lightly loaded ones to reduce controllers' energy consumption (Priyadarsini et al., 2020).

The authors in (Stojkovic et al., 2024) propose selecting the optimal CPU frequency for each serverless function by predicting its processing time and energy consumption across different frequency levels. Their results show that many functions can meet Service-Level Objectives (SLOs) at lower CPU frequencies, indicating that frequency-unaware resource allocation can cause significant energy waste.

In contrast, the present work focuses on integrating serverless computing into SDN while introducing a novel approach for efficiently provisioning sufficient resources to serverless functions in order to process incoming flows without adversely affecting service quality.

3 METHODOLOGY

3.1 Data Flow's Energy Consumption

The energy consumption and execution time of serverless functions can be predicted with high accuracy when ML models are trained on relevant input features (Stojkovic et al., 2024; Jia and Zhao, 2021). In SDN environments, serverless applications typically operate as network functions, so analysing the features of their input, which is received as data flows, and understanding its relations with the energy consumption and processing time of the corresponding serverless functions is essential for achieving energy-aware mapping.

For the purpose of training ML models based on this correlation, only a small number of studies have addressed this specific problem—that is, the relationship between flow-level features and the energy consumption or execution time of serverless applications. In (Alhindi and Djemame, 2024), the authors proposed a methodology for estimating the energy consumed by a serverless function when processing data flows. Their approach is based on analysing flows as they arrive at the serverless application and assigning each flow a weight derived from its packet count and packet sizes. The total measured energy consumption for all flows is then distributed proportionally based on these weights to estimate per-flow energy usage. However, their method does not provide a mechanism for measuring the processing time of each individual flow.

To address this limitation, we propose an alternative method for training ML models based on precise measurement of both energy consumption and processing time for each individual flow. The method consists of the following steps:

Flow extraction: The SDN traffic to be analysed

is divided into a set of single-flow Packet Capture (PCAP) files, where each file contains only one data flow.

Controlled replay: These single-flow PCAP files are subsequently replayed—either sequentially or in carefully designed batches to be captured and processed by a selected serverless application on the applications layer of SDN, while the energy and time consumed for processing each flow is monitored.

Although replaying flows individually provides the highest measurement accuracy, it is time-consuming because real network traffic contains concurrent flows. To reduce measurement time, single-flow PCAP files can be replayed concurrently, provided that flows within a group are directed to different instances of the same serverless application.

3.2 Training ML Models

The extracted features of each flow are then used to train the ML models corresponding to the key metrics required in the proposed solution. The authors (Al-hindi and Djemame, 2024) have shown that the features most strongly correlated with the energy consumed by a serverless application when processing a flow include the number of packets, packet sizes, flow duration, and Inter-Arrival Time (IAT). These features, along with their derived statistical properties—such as averages, variances, and standard deviations—can be readily obtained from PCAP files using Wireshark or similar packet-analysis tools. Beyond energy prediction, these features are also effective for estimating other metrics such as processing time and CPU usage, which will be verified in the Experiment section.

In the proposed design, each deployed serverless application must maintain its own dedicated set of three trained models on every node on which it can be executed. These models are built to predict three required metrics for each single flow:

1. **Energy consumption model:** to predict the amount of energy required for a serverless application to process a flow on a node.
2. **Processing time model:** to predict the execution latency of processing a flow by a serverless application on a node.
3. **CPU usage model:** to predict the number of CPU cores required by a serverless application to process a flow on a node.

Consequently, if multiple applications run across multiple nodes, a separate set of these three models is required for each application–node pair (e.g., 2 applications across 3 nodes result in 6 model sets). Each

node stores only the models trained using data collected from flows processed on that node, ensuring that the predictions are tailored to its hardware characteristics. This node- and application-specific training is essential because energy consumption, latency, and resource usage differ across both applications and hardware profiles. Since the number of serverless applications in SDN environments is often limited by controller capacity (Zhu et al., 2020; Xiong et al., 2016), maintaining a dedicated set of models for each application remains feasible.

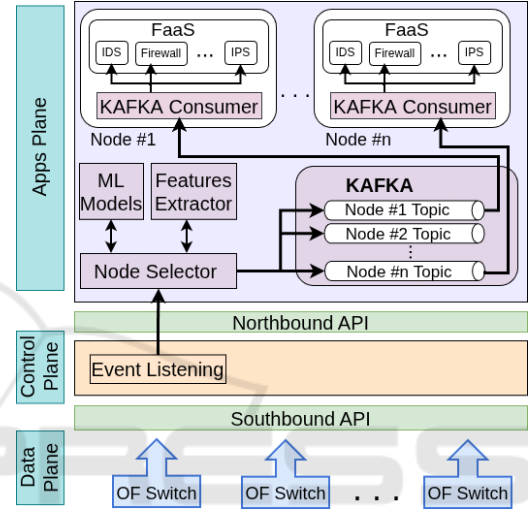


Figure 1: The proposed energy-aware architecture for mapping data flows.

3.3 Mapping Data Flows

Each deployed serverless application can register its interest in specific categories of flows—for example, TCP flows, UDP flows, or flows with particular protocol characteristics. This registration mechanism is used to determine which applications are eligible to receive a given flow. For instance, TCP flows may be forwarded to three different applications that have declared interest in processing them, whereas UDP flows may be directed to only one application. When a flow is associated with multiple applications, a separate copy of the flow is generated for each eligible application, and each copy is treated within the system as an independent flow. Consequently, every flow—or flow copy—will have its own set of three predicted metrics: energy consumption, processing time, and CPU usage.

These three metrics form the foundation of the mapping process used to select the node on which the flow will be processed. The mapper selects the node that is predicted to require the least amount of energy to process the flow while still satisfying the

service constraints. Specifically, the predicted processing time must not exceed the actual flow duration, and the selected node must be capable of providing at least the number of CPU cores required as indicated by the CPU usage prediction. Only nodes meeting these two constraints are considered viable candidates, and among them, the node with the lowest expected energy consumption is chosen for flow processing.

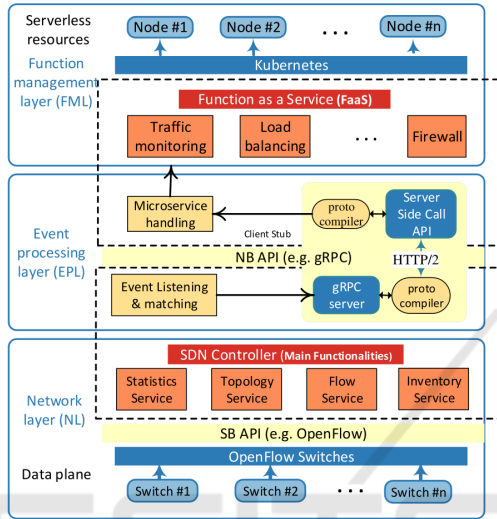


Figure 2: A serverless-SDN architecture (Banaie et al., 2025).

4 SYSTEM DESIGN

The architecture proposed in this paper (see Figure 1) is inspired by the design introduced in (Banaie et al., 2025) (see Figure 2).

In practical deployments, serverless platforms could operate across multiple heterogeneous nodes—each differing in computational power and energy consumption characteristics. Leveraging this heterogeneity to reduce the overall energy usage of serverless applications constitutes the central objective of the architecture presented in this work.

To extend the serverless-SDN architecture into an energy-aware design, several additional components are introduced (see Figure 1):

4.1 Node Selector

Algorithm 1 presents the pseudo-code of the Node Selector component that is responsible for determining the most energy-efficient execution node for each incoming data flow. The selection decision is based on the three flow-specific prediction metrics: energy consumption, processing time, and CPU usage.

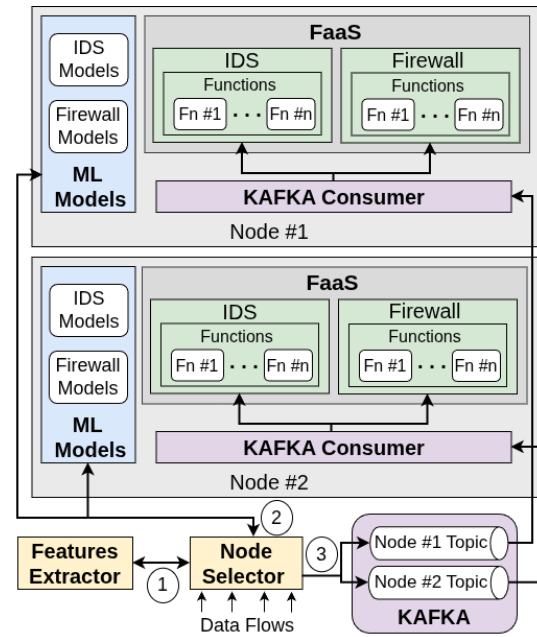


Figure 3: Implementing flow mapping.

Predictions from the ML models are generated only upon the arrival of the first packet of a flow. Once the most suitable node for processing the flow is selected, an entry is created in a hash table, where the flow ID serves as the key and the value contains the assigned node ID and the name of the target serverless application. This entry remains in the hash table for the entire lifetime of the flow and is removed immediately upon flow completion, which is detected through `FLOW_REMOVED` messages issued by the switches to the controller.

In addition to the node ID and application name, the predicted processing times for the flow across all available nodes are also stored in the hash table. This information plays an important role in latency monitoring. If a flow persists in the system for a duration significantly longer than its predicted processing time, it is flagged as experiencing abnormal latency. In such cases, the flow is immediately reassigned to the node with the shortest predicted processing time, irrespective of the associated energy cost, ensuring that service requirements and latency constraints are not violated.

4.2 ML Models

This component comprises three ML models to predict the three metrics for a given flow: energy consumption, processing time, and CPU usage. Due to hardware heterogeneity across nodes (e.g., CPU and memory differences), each node must maintain lo-

Algorithm 1: Energy-Aware Flow Mapping in SDN-Serverless Architecture.

Input : Incoming flow f with identifier flowID;
Set of applications $\mathcal{A}$;
Set of nodes $\mathcal{N}$;
App-flow interest registry I ;
ML models $\text{Models}[a, n]$ for each app-node pair;
Available CPU cores $\text{CPU_avail}[n]$

Output: Mapping (f, a^*, n^*) and updated FlowTable

Upon arrival of flow f
Extract flow features to serve as model inputs
Determine interested applications:
 $\mathcal{A}_f \leftarrow \{a \in \mathcal{A} \mid$
 $a \text{ registered for the type of } f \text{ in } I\}$
Initialize candidate set $C \leftarrow \emptyset$

for each application $a \in \mathcal{A}_f$ do
for each node $n \in \mathcal{N}$ do
Query $\text{Models}[a, n]$ with the extracted flow features to obtain:
 $\hat{E}_{a,n}$ (predicted energy)
 $\hat{T}_{a,n}$ (predicted processing time)
 $\hat{P}_{a,n}$ (predicted probability of delay)
 $\hat{C}_{a,n}$ (predicted CPU usage)
if $\hat{P}_{a,n} = 0$ and $\hat{C}_{a,n} \leq \text{CPU_avail}[n]$ then
Add candidate tuple:
 $C \leftarrow C \cup \{(a, n, \hat{E}_{a,n}, \hat{T}_{a,n}, \hat{C}_{a,n})\}$

if $C = \emptyset$ then
Handle infeasible case (fallback mapping or rejection)

else
Select $(a^*, n^*, \hat{E}^*, \hat{T}^*, \hat{C}^*)$ from C such that $\hat{E}^*$ is minimal
Map f to application a^* on node n^*
Update flow table:
 $\text{FlowTable}[\text{flowID}] \leftarrow (\text{flowID}, n^*, a^*, \hat{T}^*)$

cally trained models to ensure accurate predictions and effective mapping decisions.

4.3 Kafka

In the context of the proposed architecture, Kafka is particularly suitable because serverless applications often handle data flows that may experience substantial bursts, leading to potential flooding conditions. Kafka ensures that incoming flows are reliably

queued and delivered in order, preventing packet loss and avoiding overload of serverless components.

A Kafka Consumer is deployed on each node to consume packets from its assigned Kafka topic and forward them to the corresponding serverless application via HTTP or other supported APIs. The target application is specified in the Kafka headers of each packet.

4.4 Feature Extractor

Predicting flow-level characteristics has become an important research area (Hardegen et al., 2019; Martins, 2018; Hardegen et al., 2020; Karpinski et al., 2009), with recent advances improving accuracy for tasks such as path selection and congestion avoidance. While most studies focus on flow size classification, recent work also estimates additional features such as duration, packet count, packet size, and IAT, which are required in this study.

Accordingly, this component is responsible for extracting the necessary features of each incoming flow. Feature extraction may be performed by predicting these characteristics using only the first packet of the flow, as provided by the Node Selector, or by incorporating additional early packets to improve prediction accuracy.

However, because accurate flow-feature prediction remains an open research challenge and cannot be implemented reliably using current methods, the evaluation in this study relies on an existing dataset of data flows in which the relevant flow features are already available.

5 EXPERIMENTAL DESIGN

Due to resource limitations, the proposed solution was evaluated using a two-machine serverless-SDN testbed: one with an AMD Ryzen 7 5800H (16 GB RAM, Ubuntu 22.04) hosting ONOS as the chosen controller, virtual switches, and Kafka, and another with an Intel i5-10210U (32 GB RAM, RHEL 9) running serverless applications on Knative.

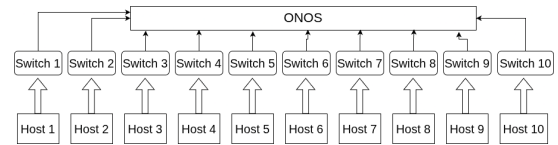


Figure 4: The topology of the virtual network.

5.1 Experimental Environment

Although many datasets provide PCAP-based traffic for experimentation, few originate from SDN environments. This study uses the InSDN dataset (El-sayed et al., 2020), which includes normal and attack traffic, where normal traffic (3.58 GB) comprises flows generated by multiple protocols (e.g., HTTP, HTTPS, SSH, DNS) from various applications (e.g., Facebook and YouTube) and network services (e.g., SSH, FTP, and Telnet).

A single serverless application was deployed to process incoming flow packets. This application performs three primary tasks: 1) comparing packet source and destination IP addresses and ports against a blacklist of prohibited values. 2) scanning packet payloads using regular expressions to detect malicious patterns. 3) repeatedly encrypting and decrypting each packet. The application is implemented in the Go programming language and deployed as a Knative service.

5.2 Training ML Models

For the experiment, only the normal traffic flows from the dataset were used. These flows are organised into 11 groups, each represented by a PCAP file containing a few thousand of individual flows. To avoid ONOS showing errors when processing large packets, all 11 groups were filtered to remove packets larger than 29,696 bytes, which represent approximately 1% of the flows.

Group 6 (2,800 flows) was selected for training due to its diversity in flow sizes and durations. Its PCAP file was split into single-flow traces using Wireshark, which was also used to extract features such as duration, packet size, IAT, packet count, and derived statistics such as averages and sums.

To replay flows in a controlled SDN-serverless environment, a Mininet topology (Fig. 4) was used to ensure each flow is captured once by ONOS, improving the accuracy of energy and processing-time measurements. The topology consists of 10 hosts and 10 switches, where each host replays a subset of flows forwarded, then, by the Node Selector to a dedicated serverless instance, resulting in 10 parallel instances. Energy consumption of each instance was measured using Kepler (Cloud Native Computing Foundation, 2024), while processing time and CPU usage were obtained from Prometheus.

To emulate the execution of the serverless application on nodes with different levels of computational capability, the cpupower tool (Silva, 2025) was used to regulate the CPU frequency of the machine

Table 1: Tested Machine Learning Algorithms and Final Hyperparameter Settings.

Algorithm	Hyperparameters (final)
Polynomial Regression	degree=2
Ridge Regression	$\alpha = 1.0$
Lasso Regression	$\alpha = 1.0$
Elastic Net	$\alpha = 1.0, l1_ratio = 0.5$
GP Regression	kernel=RBF
Random Forest	$n = 50$
KNN Regression	$k = 5$
Gradient Boosting	$n = 100, lr = 0.1, \text{depth}=4$
XGBoost	$n = 100, lr = 0.1, \text{depth}=6$
LightGBM	$n = 100, lr = 0.1, \text{depth}=6$
SVR	kernel=RBF, $C = 1.0, \epsilon = 0.1$

hosting Knative. Four frequencies—1, 2, 3, and 4 GHz—were selected to represent four distinct node configurations. Data collection for the required metrics was therefore conducted in four stages: the CPU frequency was first set to 1 GHz, the flows were replayed, and their corresponding measurements were gathered; the process was then repeated for the remaining frequencies (2, 3, and 4 GHz). For every CPU-frequency stage, a dataset is generated containing measurements of energy consumption, processing time, and CPU usage for all flows. This procedure ensures that the training data for the ML models capture behaviour across multiple CPU-performance levels, which dominate the energy consumption characteristics of the system.

5.3 Evaluation

A total of 11 ML algorithms were evaluated (Table 1), with an 80/20 train-validation split using a fixed random seed. Ridge Regression achieved the best performance for predicting energy consumption and processing time ($R^2 > 0.95$ with lowest RMSE), while Random Forest Regression performed best for CPU usage. Random Forest Regression achieved $R^2 \approx 0.60$ at 1 GHz, improving to ≈ 0.80 at higher frequencies.

To evaluate the proposed architecture, three additional flow groups from the InSDN dataset—Group 4, 5 and 8—were selected for testing. The same CPU-frequency settings used during training were also applied during evaluation. For each frequency level, Kepler was used to measure the energy consumption of the serverless application, while the Node Selector forwarded only the flows designated for that particular frequency. As a result, some flows were processed at 1 GHz, others at 2 GHz, 3 GHz, or 4 GHz, depending on their assigned configuration.

After all flows were processed, the energy-consumption measurements collected across the four frequencies were combined into a single CSV file. This file was then compared against a baseline CSV

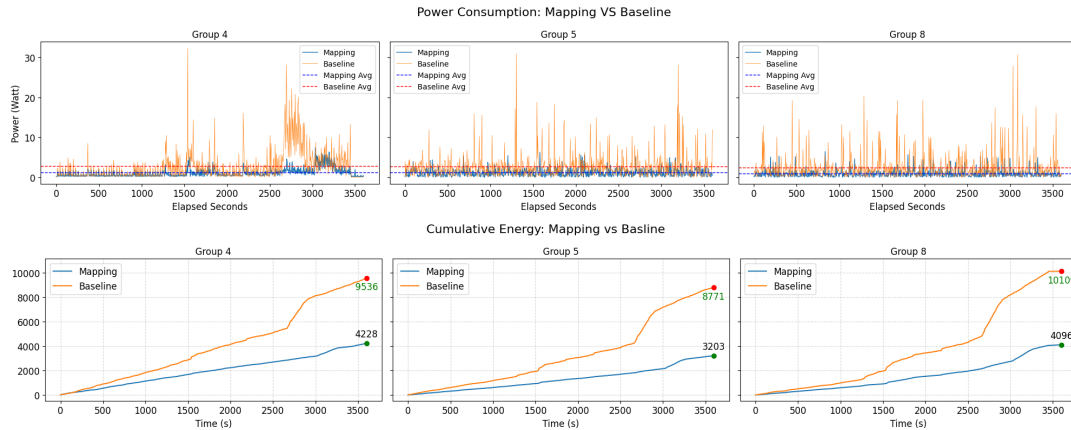


Figure 5: Comparison of Power and Energy Consumption.

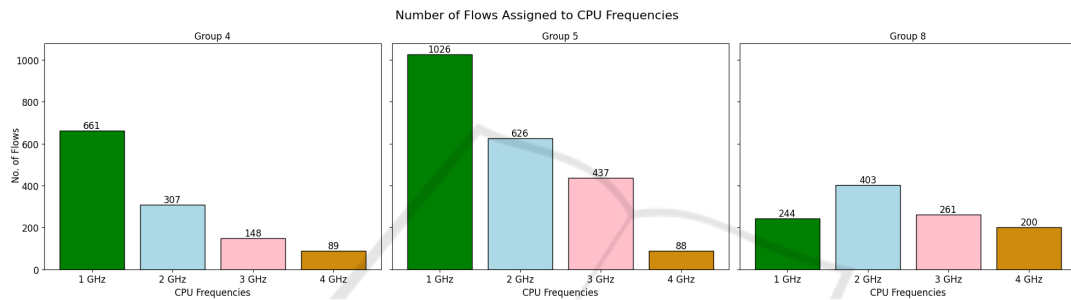


Figure 6: Flow Distribution across Selected CPU Frequencies.

file containing the energy consumption of the same flows when processed without the proposed solution. The baseline configuration refers to a setup in which all flows are forwarded by the Node Selector directly to the serverless application deployed on Knative, without passing through Kafka or invoking the node-selection process based on CPU frequency, which follows the methodology proposed in (Banaie et al., 2025). In this baseline, the CPU frequency of the machine running Knative is not explicitly controlled using the `cpupower` tool, allowing the processor to operate within its default dynamic range, from 0.4 GHz to 4.46 GHz.

Figure 5 compares the proposed solution with the baseline in terms of power and energy consumption. The baseline shows frequent spikes exceeding 30 W, while the proposed solution maintains stable power below 8 W across all flow groups. This is because the baseline allows the CPU to operate at maximum frequency under concurrent load, even when lower frequencies would suffice.

In contrast, the proposed mapping solution decomposes this workload by distributing flows across execution nodes, represented in this experiment by different CPU frequencies, according to their computational requirements. This flow-aware distribution is

the primary reason for the observed stability in power consumption. The second plot compares cumulative energy consumption over the experiment duration and shows that the proposed solution reduces total energy consumption by approximately 50% compared to the baseline. Finally, Figure 6 illustrates how flows are distributed across CPU frequencies, which represent virtual nodes in this experiment. The results indicate that the majority of flows across all three groups are processed at lower frequencies (1–2 GHz), explaining the observed reduction in energy consumption relative to the baseline configuration.

6 CONCLUSIONS

This paper presented an ML-driven, flow-aware architecture for energy-aware resource provisioning in serverless SDN environments. By predicting flow-level energy consumption, processing time, CPU usage, and delay probability, the proposed solution enables intelligent mapping of incoming data flows to execution nodes that minimise energy usage while satisfying service constraints. Unlike existing approaches that focus primarily on architectural integration or static energy optimisations, this work

introduces a practical and adaptive mechanism for energy-aware decision-making at the flow level. A methodology for accurately measuring per-flow energy consumption and execution time was introduced to support the training of prediction models tailored to specific applications and hardware configurations. Experimental evaluation using an SDN-serverless testbed demonstrated that the proposed approach can significantly reduce energy consumption without adversely affecting service quality.

Future work will focus on extending the architecture to support online model adaptation, incorporating additional flow features extracted from early packets, and evaluating scalability in real and larger multi-node SDN-serverless deployments, considering real-world node heterogeneity such as differences in RAM speed, disk I/O, and network latency. Also, it will include evaluating the approach using a more diverse set of network functions (e.g., varying computational and memory intensity) to provide a more robust validation of the mapping algorithm.

REFERENCES

- Aditya, P., Akkus, I. E., Beck, A., Chen, R., Hilt, V., Rímac, I., Satzke, K., and Stein, M. (2019). Will serverless computing revolutionize nfv? *Proceedings of the IEEE*, 107(4):667–678.
- Alhindi, A. and Djemame, K. (2024). Sdn traffic flows in a serverless environment: a categorization of energy consumption. In *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, pages 447–452. IEEE.
- Alhindi, A., Djemame, K., and Heravan, F. B. (2022). On the power consumption of serverless functions: an evaluation of openfaas. In *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, pages 366–371. IEEE.
- Assefa, B. G. and Özkasap, Ö. (2019). A survey of energy efficiency in sdn: Software-based methods and optimization models. *Journal of Network and Computer Applications*, 137:127–143.
- Banaie, F. and Djemame, K. (2022). A serverless computing platform for software defined networks. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 113–123. Springer.
- Banaie, F., Djemame, K., Alhindi, A., and Kelefouras, V. (2025). Energy efficiency support for software defined networks: a serverless computing approach. *Future Generation Computer Systems*, page 108121.
- Cloud Native Computing Foundation (2024). Kepler. <https://sustainable-computing.io>. Accessed:December 2025.
- Djemame, K. (2021). Energy efficiency in edge environments: a serverless computing approach. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 181–184. Springer.
- Djemame, K. (2022). Serverless computing: Introduction and research challenges. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 15–23. Springer.
- Elsayed, M. S., Le-Khac, N.-A., and Jurcut, A. D. (2020). Insdn: A novel sdn intrusion dataset. *IEEE access*, 8:165263–165284.
- Hardegen, C., Pfülb, B., Rieger, S., and Gepperth, A. (2020). Predicting network flow characteristics using deep learning and real-world network traffic. *IEEE Transactions on Network and Service Management*, 17(4):2662–2676.
- Hardegen, C., Pfülb, B., Rieger, S., Gepperth, A., and Reißmann, S. (2019). Flow-based throughput prediction using deep learning and real-world network traffic. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE.
- Jia, X. and Zhao, L. (2021). Raef: Energy-efficient resource allocation through energy fungibility in serverless. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 434–441. IEEE.
- Karpinski, S., Gilbert, J. R., and Belding, E. M. (2009). Predicting network flow behavior from five packets. *GSWC 2009*, page 7.
- Maity, I., Dhiman, R., and Misra, S. (2022). En-place: Energy-aware network partitioning for controller placement in sdn. *IEEE Transactions on Green Communications and Networking*, 7(1):183–193.
- Martins, R. A. S. (2018). Predicting traffic flow size and duration. Master's thesis, Universidade NOVA de Lisboa (Portugal).
- Oliveira, T. F., Xavier-de Souza, S., and Silveira, L. F. (2021). Improving energy efficiency on sdn control-plane using multi-core controllers. *Energies* (19961073), 14(11).
- Priyadarsini, M., Kumar, S., Bera, P., and Rahman, M. A. (2020). An energy-efficient load distribution framework for sdn controllers. *Computing*, 102(9):2073–2098.
- Silva, A. (2025). cpupower-gui: A graphical program to configure cpu frequency settings. <https://github.com/vagnum08/cpupower-gui>. Accessed:December 2025.
- Stojkovic, J., Iliakopoulou, N., Xu, T., Franke, H., and Torrellas, J. (2024). Ecofaas: Rethinking the design of serverless environments for energy efficiency. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 471–486.
- Xiong, B., Yang, K., Zhao, J., Li, W., and Li, K. (2016). Performance evaluation of openflow-based software-defined networks based on queueing model. *Computer Networks*, 102:172–185.
- Zhu, L., Karim, M. M., Sharif, K., Xu, C., Li, F., Du, X., and Guizani, M. (2020). Sdn controllers: A comprehensive analysis and performance evaluation study. *ACM Computing Surveys (CSUR)*, 53(6):1–40.