International Journal of Communication Systems

WILEY

RESEARCH ARTICLE

Dynamically Adaptive RSUs in VANETs: An Approach Focused on Sustainability

Luis Guilherme Silva[1] | Vandirleya Barbosa[1] | Israel Cardoso[1] | Melissa Alves[1] | Lucas Silva Lopes[1] | Paulo A. L. Rego[2] | Francisco Airton Silva[1]

[1]Federal University of Piauí (UFPI), Picos, Piauí, Brazil | [2]Federal University of Ceará (UFC), Fortaleza, CE, Brazil

Correspondence: Francisco Airton Silva (faps@ufpi.edu.br)

Received: 21 July 2025 | **Revised:** 30 August 2025 | **Accepted:** 16 September 2025

Funding: This study was supported by grants from CNPq.

Keywords: autoscaling | energy efficiency | petri nets | validation | vanets

ABSTRACT

Vehicular Ad Hoc Networks (VANETs) are increasingly pressured by applications that require high computational performance and energy efficiency from Roadside Units (RSUs), especially in dynamic and unpredictable traffic conditions. This work proposes an adaptive autoscaling model for RSUs, aiming to help system architects reduce energy consumption and carbon emissions without compromising service responsiveness. The model is developed using Stochastic Petri Nets (SPNs) and integrates a fault-tolerant reinstantiation mechanism. It was experimentally validated, proving the statistical agreement between simulated and real response times. In addition, a sensitivity analysis was performed through Design of Experiments (DoE) to identify the main energy impact parameters. The strategy achieved up to 20% lower energy consumption. In the autoscaling case study, total energy stabilized at 7.5 kWh versus 9.0 kWh without autoscaling, and the carbon footprint at 0.78 vs. 0.90 CO_2/h. Model validation showed MRT within the experimental 95% CIs across all loads ($p > 0.05$; max. Absolute deviation 5.4%). DoE indicated container backlog (C_R = 0.021) and failure probability (P_F = 0.019) as the strongest effects, with the C_R×Q_S interaction = 0.017. The proposed model effectively balances sustainability and performance in VANETs, providing a reliable basis for planning adaptive, resilient, and energy-aware vehicular infrastructures.

1 | Introduction

The evolution of VANETs has driven significant advances in communication between vehicles and infrastructure, promoting greater safety, traffic efficiency, and sustainability. In this context, RSUs play an essential role in distributed real-time data processing, ensuring the continuity of vehicular services such as driver assistance, traffic control, and emergency support [1]. However, the increasing integration of complex applications, such as environmental monitoring and logistics optimization, poses challenges to RSUs' computational capacity, which needs to balance performance and energy efficiency (EE). Thus, it is essential to develop scalable and adaptive solutions that allow dynamic adjustment of resource allocation, ensuring operational resilience in the face of unpredictable variations in processing load [2].

The operation of RSUs is subject to unpredictable variations in computational load. This results from extraordinary events such as fluctuations in traffic volume, accidents, adverse weather conditions, or sudden spikes in the data flow generated by connected vehicles [3]. The absence of adaptation mechanisms in these scenarios can lead to congestion of the computational infrastructure, degradation of service performance, and waste of resources when demand decreases. Thus, autoscaling strategies become essential by allowing RSUs to dynamically adjust their processing capacity as needed, ensuring a balance between EE and system responsiveness. Thus, the adoption of adaptive

approaches not only mitigates the impacts of overload and underutilization but also contributes to the stability and sustainability of operations in VANETs [4].

The implementation of large-scale VANETs poses considerable technical and financial challenges. Evaluating new architectures and autoscaling strategies in physical environments requires high investments in infrastructure, equipment, maintenance, and continuous monitoring [5]. These costs are further exacerbated by the absence of efficient adaptation mechanisms, which may lead to RSU overload, communication failures, and degradation of service quality [6]. Given these constraints, computational modeling presents a cost-effective and scalable alternative for analyzing system behavior under various traffic conditions without the need for expensive real-world deployments. In this context, Stochastic Petri Nets (SPNs) emerge as a powerful tool for modeling autoscaling dynamics in RSUs [7]. Their formalism allows detailed representation of critical variables such as request concurrency, resilience to instantiation failures, and system adaptation to unpredictable traffic fluctuations. By enabling the evaluation of system performance under diverse scenarios, SPN-based models contribute to optimizing resource allocation, minimizing inefficiencies, and enhancing the operational stability of vehicular communication infrastructures [7].

This study builds on two previous studies that explored the use of SPNs for modeling and optimizing dynamic computing systems. In the first study, SPNs were used to predict the behavior of an autoscaling mechanism in cloud computing environments, allowing a detailed analysis of resource allocation under different operating conditions [8]. In the second study, the approach was applied to the context of VANETs, enabling the assessment of the impact of autoscaling on system performance in vehicular networks [9]. Based on these works, the present study adapts this strategy to VANETs, expanding the scope of the modeling by incorporating EE and sustainability metrics into RSUs. To ensure the accuracy and reliability of the proposed model, a validation process was conducted using real experimental data. The results were statistically compared using the mean response time (MRT) as the main metric.

The central objective is to develop an adaptive solution that optimizes container allocation and reduces energy consumption, minimizing waste without compromising the quality of service. Thus, the proposed model improves the computational efficiency of RSUs and contributes to the sustainability of vehicular communication infrastructure, aligning VANETs' operation with the requirements of smart and environmentally responsible mobility. Thus, the main contributions of this research are:

- **Development of a Dynamic and Sustainable SPN Model**: We present a Stochastic Petri Net (SPN) model that captures the dynamic behavior of container-based RSU infrastructures in VANETs. The model integrates a reactive autoscaling mechanism capable of adjusting computational capacity based on real-time traffic demand, thereby promoting energy savings and preventing underutilization or overload. This adaptive strategy not only improves system responsiveness but also aligns with the goals of sustainable and resource-aware vehicular networks.
- **Integration of Fault-Tolerant Reinstantiation Logic**: To enhance the robustness of RSU operations, the proposed model includes a reinstantiation module that detects and recovers from instantiation failures. By modeling failure probabilities and corrective actions, the SPN ensures continuity of services even under adverse conditions. This feature reinforces operational resilience and reduces energy waste caused by repeated resource allocation attempts, contributing to a more dependable and efficient infrastructure.
- **Experimental Validation with Empirical Load Scenarios**: The accuracy of the proposed SPN model was validated through real-world experiments using the Pasid-Validator tool. The autoscaling policy was implemented in a physical environment and tested under varying traffic loads. Model predictions were statistically compared with empirical MRT data, and results showed consistent alignment within a 95% confidence interval. This validation confirms the model's applicability for performance forecasting and supports its use in practical deployment planning.

The remainder of this paper is organized as follows: Section 2 introduces the foundational concepts necessary for understanding this study. Section 3 reviews related work in the literature. Section 4 presents the methodological framework adopted, including the overall research process and the system architecture. Section 5 presents the proposed SPN model. Section 6 discusses the validation of the model. Section 7 describes the results obtained from the experimental design. Section 8 showcases two case studies to illustrate the application of the model. Finally, Section 9 summarizes the main findings and outlines directions for future research.

2 | Background

This section presents essential concepts for understanding the main components of this study. Initially, concepts about SPNs are presented. In addition, the functioning of Design of Experiments (DoE) and how its results are generally interpreted are discussed.

2.1 | Petri Nets

SPNs have transitions that can be immediate or timed [10]. Timed transitions have an exponentially distributed delay, while immediate transitions have an associated delay time equal to zero [11]. SPNs allow the modeling and probabilistic analysis of systems. The memorylessness property of the exponential distribution in the firing delay implies that SPNs are isomorphic to continuous-time Markov chains (CTMCs), thus providing performance and dependability measures [12]. It is essential to highlight that we are using the Generalized Stochastic Petri Net (GSPN), which is an extension of the SPN [13]. GSPNs are suitable for representing discrete event dynamic systems because they have different types of timed transitions. However, for the sake of simplification, we will call them SPNs in the text.

Figure 1 presents a simple example of an SPN and its main components. Arcs (directed edges) connect places to transitions and vice versa. Tokens can reside at places that denote the state (i.e., marking) of an SPN. Tokens typically represent new

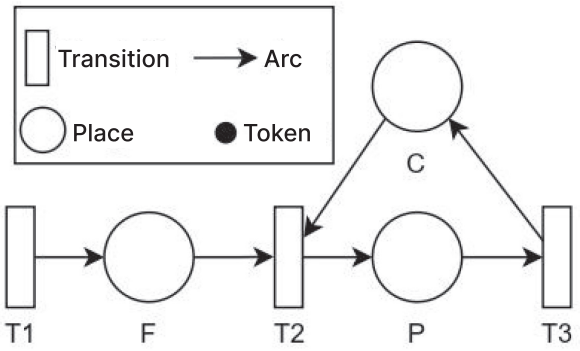


FIGURE 1 | Illustrative example of a Stochastic Petri Net (SPN), highlighting its main components: places (circles), transitions (rectangles), directed arcs, and tokens (dots). The model represents the flow of tasks through request generation, queuing, processing, and completion with capacity control.

requests entering the system. In the present work, each token arriving in the system represents a new packet to be delivered to the destination. The behavior of an SPN is defined in terms of a token flow. The performance of an action or event (triggering of a transition) in the system is linked to preconditions.

Tokens are created and destroyed according to transition triggers. Each transition has an associated time that follows a particular probability distribution. In the example in Figure 1, new tokens are generated in transition T1 that follows a specific arrival rate. The tokens are then queued in place F, which does not have a maximum queuing limit in this case. To actually "enter" the system, the token must go through a time linked to transition T2. This entry will only occur if there is sufficient capacity in place C. C represents the maximum parallel capacity of the system. When the token reaches place P, transition T3's time begins to count, and then a cycle ends.

2.2 | Sensitivity Analysis

Sensitivity analysis examines the impact of specific input data on results, identifying potential flaws in computational systems. Techniques are then applied to optimize these systems in different scenarios [14]. In general, system designers use sensitivity analysis to assess how much a metric is affected by changes in the model [15]. There are several ways to perform this analysis, including regression analysis, perturbation analysis, Monte Carlo simulation, parametric differential analysis, correlation analysis, and percentage difference.

Choosing the appropriate method can be challenging, and it is necessary to consider the available computational resources and the characteristics of the problems addressed. Sensitivity analysis can provide the necessary security and direct the results according to the perspective previously established by the system administrators. In this work, we apply sensitivity analysis techniques based on DoE and Percentage Difference.

DoE is a set of statistical techniques that improve the understanding of the product or process in question. DoE is a very effective technique for exploring new processes and gaining

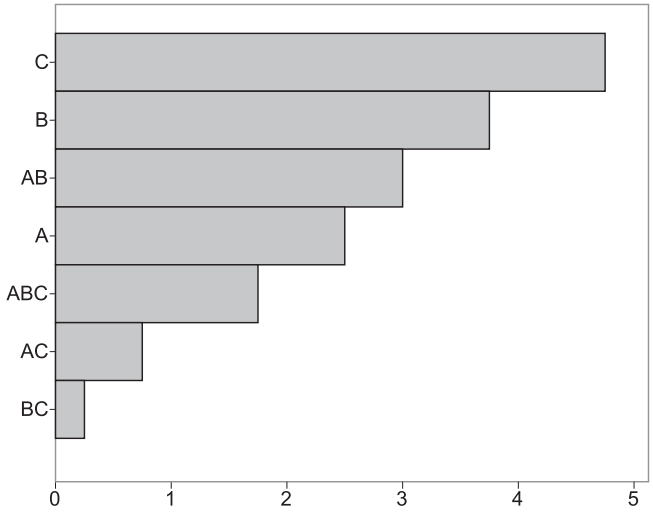


FIGURE 2 | Generic factor effect graph used in Design of Experiments (DoE). The bars are sorted in descending order of impact, helping identify which factors most influence the target metric.

a deeper insight into existing processes, followed by optimizing these processes to achieve world-class performance. In the specialized literature [15–17], categories of graphs are found that are frequently used in experiments with the DoE approach. The factor effect graph, represented by bars arranged in ascending order, emphasizes the relative impact of each factor. The larger the bar, the greater its impact, allowing a clear understanding of each factor's influence. Figure 2 presents a graph of the effect of the factors. The graph presents three factors: A, B, and C. Factor C is the factor with the most significant impact.

The factor effect plot allows us to identify which factor interaction has a more relevant impact on the optimization process or study design, indicating where attention should be directed. Understanding the interactions between factors allows us to select the best combination of measures, identifying patterns of cumulative or degrading effects between factors. The interaction between factors A and B can be estimated using the Equation (1). The plot {A,B(+1)} represents the effect of factor A on the highest level of factor B, while {A,B(−1)} represents the effect of factor A on the lowest level of factor B.

$$I_{A,B} = \frac{1}{2}\left(E_{A,B(+1)} - E_{A,B(-1)}\right) \tag{1}$$

The purpose of interaction graphs is to establish relationships between factors. An interaction occurs when the influence of one factor on the outcome is increased (or decreased) by changes in the levels of another factor. If the lines on the graph are parallel, this means that the factors are not related. If the lines are not parallel, this indicates a significant interaction between the factors in question. Figure 3a illustrates an example where there is no interaction between the factors, since the lines are parallel. Figure 3b illustrates an example of interaction between factors, since the lines intersect. The change in a metric for factor A at level A_1 is greater than the change at level A_2. Changes in the levels of factor A for a given metric indicate a dependency between the levels of factor A and the levels of factor B.

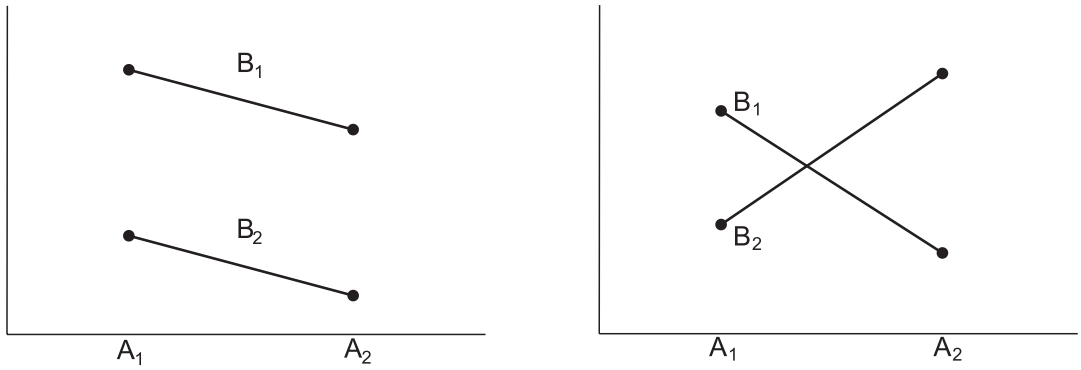


FIGURE 3 | Examples of interaction graphs between factors. (a) Illustrates low interaction between factors, with parallel lines indicating independence. (b) Shows strong interaction, where intersecting lines reveal dependencies between factor levels.

TABLE 1 | Related works.

Work	Metrics	Energy	Autoscaling	Evaluation method	Focus
[18]	Packet delivery rate, packet latency, avg. energy consumption	✓	✓	Simulation	Communication
[19]	Throughput, packet delivery rate, communication delay, communication overhead	✗	✓	Simulation	Communication
[20]	Packet delivery rate, end-to-end delay, routing overhead, throughput, energy consumption	✓	✓	Simulation	Communication
[21]	Energy consumption, queue wait time, load balancing efficiency	✓	✓	Simulation	Energy efficiency
[22]	Energy consumption, % of requests served	✓	✓	Simulation	Energy efficiency
[23]	Task response time, vehicle energy, load balancing	✓	✓	Simulation, modeling	Autoscaling optimization
[24]	Residual energy, end-to-end delay, delivery rate	✓	✓	Simulation	Autoscaling optimization
[25]	Workload rate, latency, load balance index	✗	✓	Simulation, modeling	Communication
[26]	Total energy, network lifetime, imbalance degree	✓	✓	Simulation, modeling	Energy efficiency
[27]	Energy consumption, network load, avg. packet delay	✓	✓	Simulation, modeling	Energy efficiency
[28]	Transmission time, packet loss, response time, throughput	✗	✓	Simulation	Communication
[29]	Deadline miss, stretch, delivery success	✗	✓	Simulation, modeling	Autoscaling optimization
[30]	Deadline miss, queue time, throughput, delivery rate	✗	✓	Simulation	Communication
This Work	Energy consumption, carbon footprint, energy efficiency, response time	✓	✓	Modeling	Sustainability

3 | Related Works

This section presents an analysis of related works, focusing on studies that address load balancing and/or energy consumption (EC) in VANETs. The works are classified into two groups based on the evaluation method employed: (i) Simulation-Based Analysis in VANETs and (ii) Model-Based and Simulated Approaches to VANETs. Table 1 presents the main characteristics of these studies.

3.1 | Simulation-Based Analysis in VANETs

The studies [18–21] used Network Simulator 2 (NS2) as the primary simulation tool. The study [18] proposed an architecture that integrates UAVs to balance network traffic in congested areas, using NS2 and BonnMotion. The study [19] developed a routing model to improve communication quality in distributed VANETs, evaluating different traffic scenarios using NS2. The study [20] focused on resource allocation and energy balancing in UAV-assisted networks, aiming to reduce latency and communication overhead through simulations in NS2. Finally, the study [21] proposed a hybrid load balancing scheme, minimizing EC in RSUs through an efficient request scheduling algorithm, evaluating its performance in NS2, SUMO, and OpenStreetMap.

On the other hand, the studies [22, 24, 28, 30] used different simulation tools. The study [22] presented an energy-efficient scheduling algorithm for RSUs powered by renewable energy sources, distinguishing between traditional and intelligent applications, with simulations conducted in NS3, SUMO, and OpenStreetMap. The study [24] proposed an adaptive load balancing (ALB) mechanism to mitigate congestion in VANETs, organizing RSUs into clusters and managing route selection and request scheduling, evaluated using VanetMobiSim with NS2 extension and CANU Mobility Simulation Environment. The study [28] developed a cooperative approach to load balancing in RSUs, improving communication efficiency between vehicles and infrastructure, evaluated in CSIM19, demonstrating a reduction in the rate of expired requests. Finally, the study [30] proposed an ALB scheme to optimize data dissemination in VANETs within the V2I model, also evaluated in CSIM19, resulting in improvements in delivery success rate and communication reliability. Although these simulation-based approaches offer valuable insights into VANET behavior, they often lack formal models capable of capturing system dynamics under autoscaling policies and energy constraints.

3.2 | Model-Based and Simulated Approaches to VANETs

The studies [23, 26] utilized bioinspired algorithm-based modeling to optimize load distribution in VANETs. The study [23] proposed a vehicular computation offloading scheme with guaranteed load balancing, employing Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) to model the dynamic task allocation within the network. Similarly, the study [26] used a hybrid ACO-ABC algorithm to optimize EE and load distribution in fog-based VANETs. Both approaches were evaluated through simulations in NS2, demonstrating improvements in load distribution and energy consumption.

Alternatively, the studies [25, 27, 29] applied statistical modeling to predict and balance the load among network elements. The study [25] introduced QualityScan, a scheme based on Little's Law to estimate the load on access points (APs) and optimize user allocation, validated through simulations in NS2. The study [27] proposed a mathematical model based on packet arrival and transmission rates, exploring different node distances to minimize energy consumption, with validation conducted in MATLAB. Meanwhile, the study [29] employed statistical methods to predict the future load of RSUs, ensuring efficient request balancing, with validation performed through simulations in CSIM19. Despite the advancements in modeling and hybrid methods, many studies rely on abstract assumptions or do not validate their models against real deployments, which limit their practical applicability.

3.3 | Contributions of This Work

Unlike existing approaches in the literature, our methodology exploits the flexibility of SPNs, offering a probabilistic and structured model for energy-efficient load balancing in VANETs. Our approach dynamically adjusts the computational resources of RSUs by integrating autoscaling systems, reducing EC and carbon emissions. This adaptability ensures that resources are allocated efficiently according to network demand, minimizing underutilization and overload scenarios. In addition, we employ the DoE methodology to perform sensitivity analyses, which allow a quantitative assessment of the main parameters that influence the system's sustainability.

This methodological improvement enables more accurate decision-making, promoting a scalable and environmentally sustainable vehicular infrastructure. MRT was used as a performance indicator to evaluate the system's responsiveness under different load levels. Finally, it is worth highlighting that the proposed model was validated through practical experimentation with the Pasid-Validator tool, demonstrating statistical adherence between the simulated behavior and the empirical data, which reinforces the reliability and applicability of the model in real scenarios.

4 | Methodological Approach

This section presents the methodological approach adopted in this study, structured to ensure clarity and reproducibility of the proposed model. First, we describe the overall methodological process, illustrated by a flowchart, which outlines the sequential stages of the research—from literature review and architectural definition to modeling, validation, sensitivity analysis, and case studies. Next, we detail the system architecture, which represents the foundation of the SPN model and guides the evaluation of autoscaling and sustainability strategies in vehicular networks.

4.1 | Considered Architecture

This section describes the architecture developed to create the SPN model. Figure 4 illustrates the proposed architecture to integrate dynamic scaling and energy sustainability in vehicular

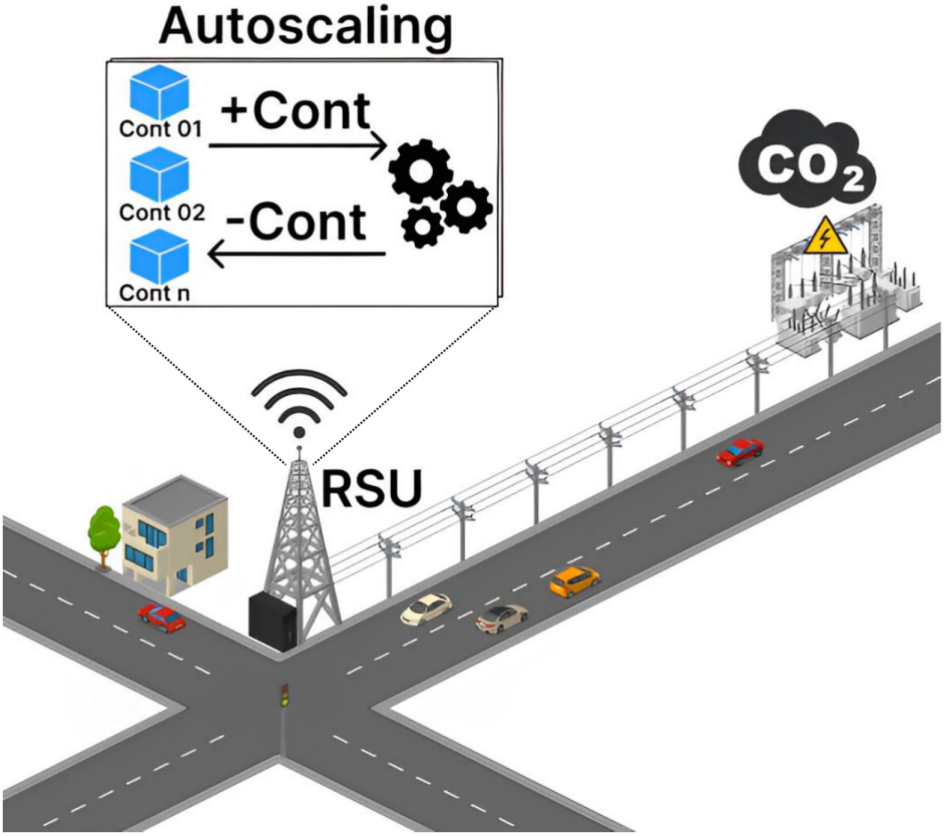


FIGURE 4 | Proposed architecture for adaptive RSUs with autoscaling and energy-aware operation in VANETs. The system reacts dynamically to variations in traffic volume, adjusting the number of active containers to maintain performance and minimize resource waste.

environments with RSUs. The system is designed to operate adaptively, responding to changes in vehicular traffic volume over time. The central component is the RSU unit, which has elastic capacity to meet different levels of load generated by connected vehicles. As the flow of vehicles increases, message generation intensifies, resulting in a greater number of requests that need to be processed by the RSU. To avoid overload and ensure quality of service, the system incorporates an autoscaling mechanism that allocates additional containers as demand grows. Allocation occurs in an automated and reactive manner, ensuring that the infrastructure dynamically follows traffic demands without resorting to permanent oversizing.

The architecture is based on the principle that the volume of traffic on highways varies significantly throughout the day, directly influencing the rate at which vehicles generate data. During periods of low traffic, the system operates with a reduced number of containers, consuming only the resources necessary to maintain service responsiveness. However, during peak times, the infrastructure needs to respond quickly to increased computational load, avoiding bottlenecks that compromise V2I communication performance. To this end, the autoscaling mechanism is activated reactively whenever the occupancy of the request queue exceeds a certain threshold. This threshold, set at 80% of the queue capacity, was defined based on empirical tests and represents a critical inflection point that precedes saturation. This preventive policy allows new containers to be instantiated before the system reaches an overloaded state, maintaining a balance between performance and energy consumption.

The process of allocating and deallocating containers occurs from two logical repositories: the idle container pool and the set of active containers in operation. When the autoscaling policy identifies the need for expansion, new containers are instantiated from the pool and added to the RSU processing capacity. Similarly, when the request load decreases and underutilized instances are identified, the system controls the removal of excess containers and returns them to the pool. This strategy prevents computational resources from remaining unnecessarily active, which is essential to reduce EC without compromising service continuity. The scaling logic is also sensitive to failures during the instantiation process. To address this, the reinstantiation module is used, which automatically detects and corrects failures, ensuring infrastructure resilience.

4.2 | Overall Methodology

The methodological process adopted in this study was structured in sequential stages, ensuring consistency between conceptual foundations, formal modeling, empirical validation, and case-based application. Figure 5 illustrates the main phases followed throughout the research.

- **Literature Review:** The first stage consisted of a systematic review of works addressing autoscaling, EE, and performance optimization in VANETs. This step allowed identifying the main strategies employed in the state of the art, as well as their limitations regarding sustainability

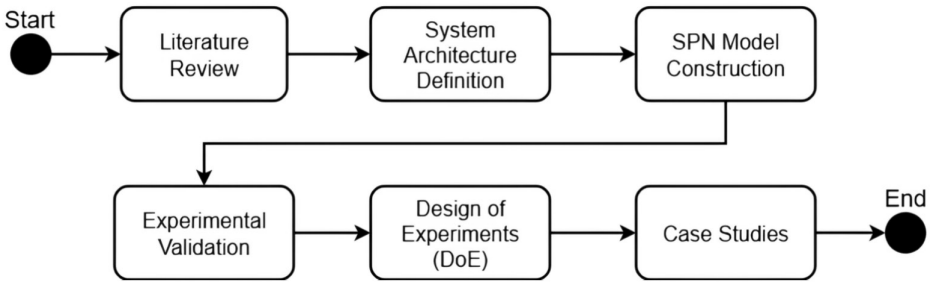


FIGURE 5 | Overall methodological framework adopted in this study, comprising the sequential stages: literature review, system architecture definition, SPN modeling, experimental validation, DoE-based sensitivity analysis, and case studies.

metrics. The review also highlighted the potential of SPNs as a formal tool for modeling adaptive systems, guiding the definition of the present proposal.

- **System Architecture Definition:** Based on the insights obtained from the literature, an adaptive RSU architecture was specified, integrating dynamic autoscaling and reinstantiation mechanisms. This architecture was conceived to reflect real vehicular environments, where computational demand fluctuates with traffic intensity and where failures must be efficiently handled to maintain service continuity. The architecture served as a blueprint for the SPN model.
- **SPN Model Construction:** The proposed architecture was formally represented through SPNs, enabling a probabilistic analysis of system behavior. The model was structured into four modules: Admission, RSU Processing, Autoscaling, and Reinstantiation. Each module was designed to capture the stochastic dynamics of request arrivals, resource allocation, fault detection, and recovery, thus providing a detailed yet analytically tractable representation of the system.
- **Experimental Validation:** To ensure that the proposed model accurately reflected real operational dynamics, an empirical validation process was conducted. Using the Pasid-Validator tool, execution times were collected in a controlled environment, instrumented with a request generator and a load balancer node. These data were used to parameterize the SPN transitions, allowing simulation results to be statistically compared with experimental Mean Response Times (MRTs). This stage confirmed the representativeness and fidelity of the model.
- **Design of Experiments (DoE):** Once validated, the model was employed in a sensitivity analysis using DoE. This methodology provided a structured way to evaluate the influence of critical factors—such as the number of containers, reserved capacity, and failure/reinstantiation probabilities—on EC and efficiency. By applying statistical techniques, DoE enabled quantifying the relative impact of each factor and their interactions, offering deeper insights into the trade-offs between performance and sustainability.
- **Case Studies:** Finally, two case studies were designed to demonstrate the applicability of the model. The first compared system behavior with autoscaling enabled and disabled, highlighting the energy and environmental gains of adaptive mechanisms. The second analyzed the effect of varying the number of containers, evidencing the trade-off between scalability and EE. These case studies illustrate the practical relevance of the model for system planning and optimization in VANET scenarios.

This methodological sequence ensured that the study progressed in a rigorous and coherent manner: from the identification of theoretical gaps, through the construction of a formal and validated model, to its application in scenarios that demonstrate both technical performance and sustainability implications.

5 | Model

This section details the developed model, structured in four main modules, which represent the system operations: (1) Admission, (2) RSU Unit, (3) Autoscaling System, and (4) Reinstantiation System. These modules were designed to analyze the system's performance in relation to traffic dynamics and computational demand while respecting the parameters of the adopted architecture.

5.1 | Model Structure

Figure 6 illustrates the main structure of the model. The Admission module controls the initial flow of requests. In it, place `Tad` represents the incoming requests generated by traffic, while place `Pqe` organizes the transfer of requests to the next stage. The RSU module then processes pending requests. Queued requests are stored in place `Ppqe` until sufficient resources are available in place `Pcaprsu`, indicating containers ready for execution. The transition `Trsu` transfers the request to place `Pprsu`, where processing occurs. After processing, the transition `Ttsrsu` returns the containers to place `Pcaprsu`, allowing their use in new operations.

The autoscaling module dynamically manages the system's capacity based on demand. When a high queue of requests and capacity reduction is detected, transition `T4` is activated, triggering the instantiation of new containers from location `Pcres`. These containers pass through location `Pic`, where the time required for instantiation is simulated by transition `Tic`, before being sent to location `Pcaprsu` via transition `T5`. Conversely, when demand decreases, idle containers are released by transition `T2`, temporarily stored in `Pretc`, and reallocated to the initial reservoir `Pcres` via transition `T3`.

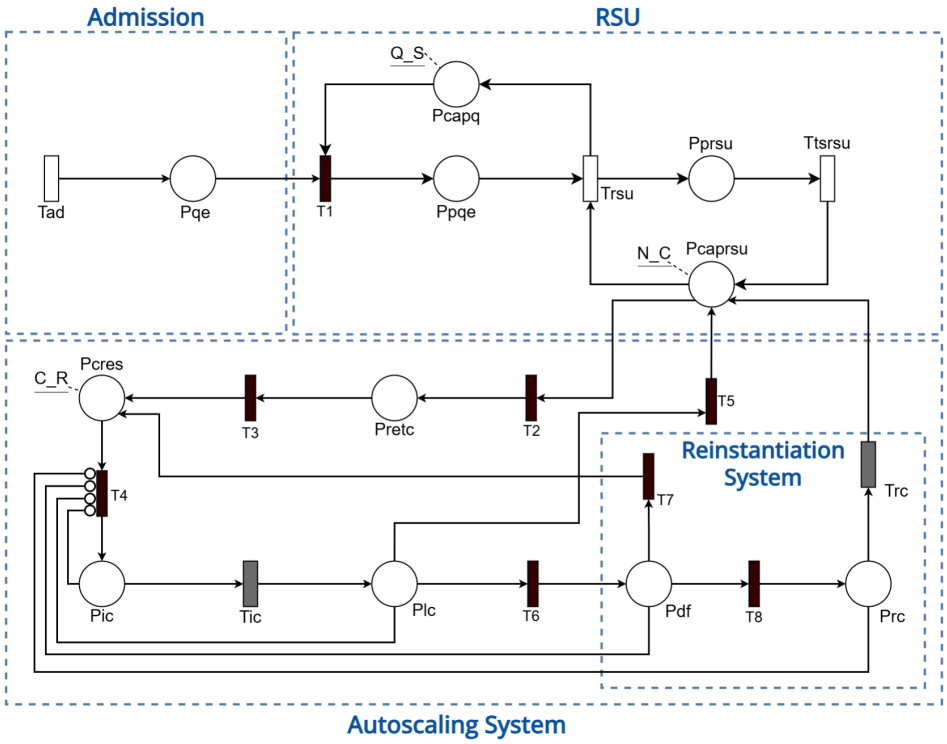


FIGURE 6 | SPN model representing the dynamic behavior of RSUs. The model comprises four modules: Admission, RSU Processing, Autoscaling, and Reinstantiation, interconnected to simulate real-time response to traffic load fluctuations and failures.

Finally, the Reinstantiation module handles failures during the instantiation process, ensuring the system operates continuously. When a failure occurs, transition `T6` redirects the container to the detection location `Pdf`, which tries to correct the failure. If successful, the container proceeds to the `T8` transition. If the failure is not resolved, the `container` returns to the initial reservoir `Pcres` via transition `T7`. If the failure is corrected, transition `T8` forwards the container to location `Prc`, where the deterministic transition `Trc` finalizes the process by reallocating the container to the capacity location `Pcaprsu`. This approach strengthens the resilience and operational efficiency of the system. The model employs exponential distributions for all timed transitions, consistent with the classical semantics of SPNs. This choice ensures analytical tractability and allows rigorous derivation of steady-state measures. Although phase-type or general distributions could provide a more detailed representation of bursty vehicular traffic, the exponential abstraction was adopted to balance realism with mathematical solvability.

5.2 | Model Metrics and Guard Conditions

In the context of this work, the metrics are focused on evaluating the energy impact of the system. The metrics used in the experiments were EC, Carbon Footprint (CF), and EE. Each of these metrics is essential to measuring performance in terms of resource consumption and environmental impact, considering the system's behavior under different loads and operating conditions.

Equation (2) contains the EC metric, which aims to calculate the total energy used by the active components of the system during operation. To do this, the calculation takes into account the energy consumed by both the RSU and the virtualization container (Cont), considering the total operating time. The calculation is based on the system utilization, checking the expectation of having tokens in `Pprsu` and `Pcaprsu`, that is, requests being processed by the RSU. `E_OP_RSU` is the energy required to keep the RSU active, and `E_OP_Cont` is the energy required to keep the container running. The factor `Time` represents the duration of time that these components are active and processing information. The calculation considers the total energy consumed per hour by the components while they are operating simultaneously.

$$\mathbf{EC} = ((E\{\#Pprsu\} \times E\_OP\_RSU + E\{\#Pcaprsu\} \times E\_OP\_CONT) \times Time) \quad (2)$$

The metric that calculates the CF, contained in Equation (3), calculates the environmental impact of the system's energy consumption, considering the carbon emission factor (CEF). This factor depends on the energy source used, since different energy sources (such as coal, gas, or renewable energy) have different Carbon Dioxide (CO_2) emission intensities. The difference between this metric and Energy Consumed is that the CF takes into account the environmental impact of the energy used, multiplying the total energy consumed by the CEF. The CEF is a constant that reflects the amount of CO_2 emitted per unit of energy consumed and is essential for verifying greenhouse gas emissions. This makes the CF a metric that assesses the sustainability of the system in terms of environmental impact. It should be emphasized that the adopted EC model assumes linear proportionality between the number of active tokens (requests/containers) and the energy

used. This abstraction was chosen to simplify the stochastic representation, which may play a relevant role in real-world RSU deployments.

$$\mathbf{CF} = CEF \times EC \tag{3}$$

Equation (4) covers EE and is based on the relationship between completed tasks, represented by the number of successfully processed tasks, and the energy consumed. This calculation seeks to reflect the efficiency of the system in using its energy resources to perform operations. Based on the system throughput, the metric is a proportion between the requests being processed, the service time, and the energy consumption. It is measured by the expectation of tokens in the place of RSU processing `Pprsu`, divided by the time it takes for the subsequent request to be processed `Ttsrsu`, after which it is divided by the energy consumption. Thus, it reflects the system's ability to perform its tasks using the least amount of energy possible, and is essential for evaluating the system's performance in terms of sustainability and resource optimization.

$$\mathbf{EE} = \left(\left(\frac{Pprsu}{Ttsrsu} \right) \div EC \right) \times 100 \tag{4}$$

The MRT of the system is presented in Equation (5). Usually, the MRT can be obtained from Little's Law [31], which requires system stability. That is, the request arrival rate (AR) must be lower than the processing rate. However, in this work, we adopt a metric based directly on the expected quantities of tokens (requests) observed in the stochastic model. The equation expresses the MRT as the ratio between the sum of the expected quantity of requests waiting in the RSU queue (`E{#Ppqe}`) and in processing in the RSU (`E{#Pprsu}`). This total is divided by the effective average service rate of the RSU. This rate is represented by the ratio between the expected number of requests being processed in the RSU (`E{#Pprsu}`) and the average service time of the RSU (`{Ttsrsu}`). In this context, `E{#Ppqe}` and `E{#Pprsu}` represent, respectively, the statistical expectation of tokens in the places that model requests waiting in the queue and processing. The denominator of the equation provides the RSU with an effective throughput. This value is calculated based on the RSU processor's average occupancy and the time required to complete each service. Thus, the MRT reflects the average time required for a request to travel through the system. This estimate considers both the queue dynamics and the service capacity of the RSU. The adopted metric is more appropriate to the stochastic model used, since it directly incorporates the expected values observed in the system.

$$\text{MRT} = \frac{E\{\#Ppqe\} + E\{\#Pprsu\}}{\left(\frac{E\{\#Pprsu\}}{\text{Ttsrsu}} \right)} \tag{5}$$

Table 2 details the guard conditions associated with transitions T2, T4, T5, T7, and T8, which are fundamental for the dynamic control of the system. These conditions are necessary for the activation of the autoscaling and reinstantiation mechanisms of containers. For transition T2, the guard condition is triggered when the number of idle containers in `Pcaprsu` is detected. In this case, the inactive containers are moved back to the reserved containers storage location, `Pcres`, optimizing the available resources. The T4 transition is triggered when the number of containers in `Pcaprsu` represents less than half of the initial capacity, which causes the autoscaling mechanism to be activated. However, the `SWT` variable that defines the activation must be equal to 1, thus activating autoscaling. This results in the instantiation of a new container to meet the growing demand of the system.

TABLE 2 | Guard conditions used in the model.

Transition	Guard condition
T2	((#Pcaprsu + #Pprsu) > (N_C*(N_C/2))) AND (#Pprsu <= (N_C/(N_C/2)))
T4	(SWT = 1) AND (#Pcaprsu <= (N_C/(N_C/2)))
T5	P_F
T7	1—P_R
T8	P_R

In the proposed model, P_F (failure weight) and P_R (reinstantiation weight) are routing weights assigned to the immediate transitions T5, T7, and T8, encoding probabilistic branching without inflating the state space. In transition T5, the guard condition refers to the detection of system failures: when a failure is identified, the transition is activated, and the token follows the failure path with weight P_F or the success path with 1—P_F, allowing the system to execute the necessary procedures to maintain uninterrupted operation. Transitions T7 and T8 are directly linked to the reinstantiation process: T7 is triggered when a container must be reinstantiated after a failure, while T8 is activated when the reinstantiation succeeds and the container is restored to its functional state. By concentrating heterogeneous causes of failure and recovery (e.g., transient contention, software restarts, or short-lived network issues) into compact weights, the model preserves analytical tractability while explicitly capturing the autoscaling and reinstantiation logic.

6 | Model Validation

This section describes the validation process of the proposed SPN model to represent the behavior of container autoscaling systems in VANETs. The validation seeks to verify the accuracy of the model by reproducing the real behavior of the system under different load conditions, obtained through controlled empirical experimentation. The methodological procedure was structured in stages, including the construction of the model, the instrumentation of the experiment, the collection of data, and the comparison between the simulated and experimental results. The main metric used for analysis was the MRT, which reflects the average time required for a request to travel through the system. The following presents the adopted methodology and the details. After that, the experiment's architecture, the validation tool, the comparative results, and the final discussion are presented.

6.1 | Validation Methodology

The validation methodology adopted in this work was structured in sequential steps, as illustrated in Figure 7. The process begins with system modeling, moves on to instrumentation of the experimental environment, and ends with statistical analysis of the adherence between model and experiment. Each step is described in detail below.

- **Create SPN model:** The model presented in Section 5, based on Stochastic Petri Nets, was used to represent the autoscaling system under continuous request arrival. The model structure includes queues, multiple parallel instances, and a reactive scaling policy. For validation purposes, the model parameters were adjusted to reflect the real system configurations.
- **Implementation of the autoscaling logic in Pasid-Validator:** The scaling policy was coded directly in the load balancing component of the Pasid-Validator tool.[1] The added logic monitors the queue occupancy and dynamically instantiates new services whenever the usage exceeds the 80% threshold.
- **Setup Experimental System:** The physical environment consisted of two main nodes: a source node, responsible for request generation, and a load balancer node, responsible for dynamic distribution and automatic scaling control. The model was structured to represent the specified architecture.
- **Execution of the experiment with low load:** Initially, the system was executed with a reduced request rate, in order to allow the stable collection of intermediate processing, transmission, and response times.
- **Collection and mapping of intermediate times:** During the execution with low load, Pasid-Validator recorded the times between relevant events (such as sending, receiving, and completion of the request). These average values were used to feed the model transitions directly.
- **Model feeding with experimental times:** The collected times were incorporated into the SPN model, ensuring that the simulation considered the real dynamics observed in the physical environment.
- **Model simulation with different loads:** The model was run in steady state with multiple AR configurations. For each case, the MRT and standard deviation were obtained.
- **Practical execution with load variation:** In parallel to the simulation, the real system was run under the same AR configurations. For each scenario, the experimental MRTs were collected from the Source component instrumentation.
- **Comparison between model and experiment:** The results obtained in the simulations and the physical experiments were directly compared. The adherence was verified based on the proximity between the MRT values obtained in the two domains.
- **Calculation of *p*-Value in relation to MRTs:** From the data obtained from the simulation and the experiment, the p-Value between the MRTs in each scenario is calculated.
- **Final statistical validation:** The compatibility between the results was statistically evaluated. The model was considered validated whenever the simulated MRT remained within the 95% confidence interval of the experimental observations.

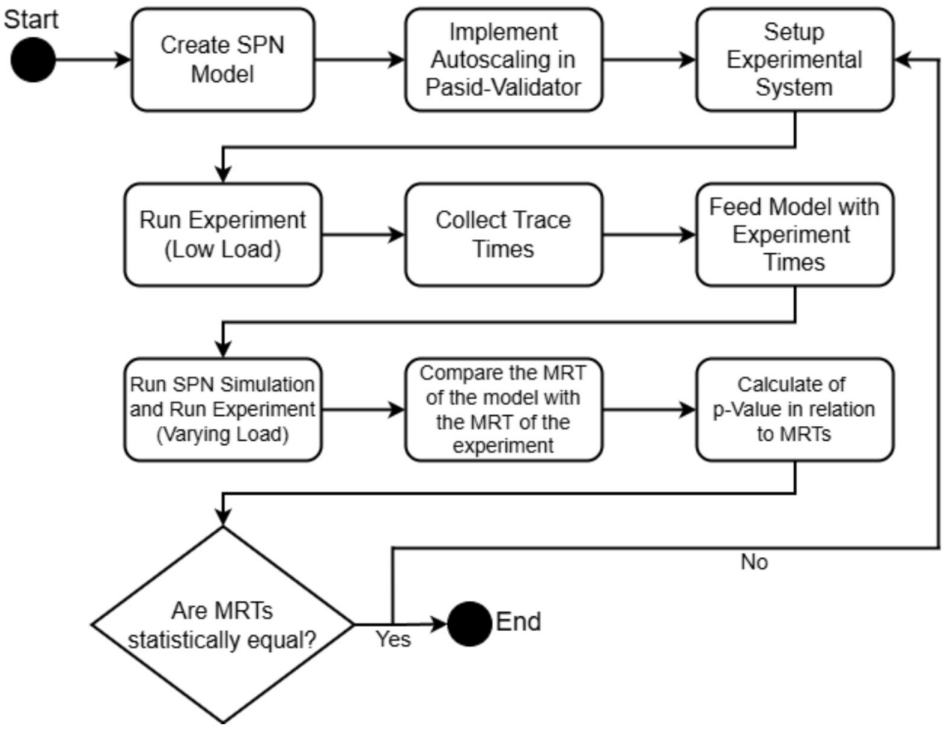


FIGURE 7 | Validation methodology for the proposed SPN model. The process includes model construction, instrumentation of the experimental environment using Pasid-Validator, empirical data collection, simulation, and statistical comparison of MRT values.

6.2 | Experiment Architecture and Model Used

The experimental architecture was designed to reproduce the main elements of the modeled system. The main focus was on the service autoscaling logic and the vehicle infrastructure based on RSUs. The physical environment consisted of two machines interconnected in a network, as illustrated in Figure 8. The Source is responsible for generating requests and simulating the demand from vehicles in the VANET. This machine emulates the behavior of mobile nodes, sending requests continuously according to the configured AR parameter. The LoadBalancerProxy represents the RSU, being responsible for the queue and for dynamically distributing requests among active services. This machine implemented the autoscaling logic, which monitors the occupancy of the request queue and instantiates new services reactively.

The scheduling policy adopted in the experiment follows the same criteria as the SPN model. Whenever the utilization of the request queue exceeds 80% [32] of CPU usage, a new service instance is automatically created. This value was chosen because it represents a saturation point close to the limit of the queue capacity, allowing the growth in demand to be anticipated without incurring excessive delays or loss of performance. Each instance is represented by an independent thread, capable of processing requests in parallel. The number of active services, therefore, varies dynamically according to the intensity of the load. The infrastructure was implemented with the help of the Pasid-Validator tool, extended with support for adaptive scheduling. This extension allows the same adaptation mechanisms represented in the model to be replicated in the physical environment, ensuring structural and behavioral coherence between the experiment and the simulation.

For validation, the SPN model presented in Figure 6 (Section 5) was used. To ensure equivalence with the experimental environment, the model parameterization values were adjusted to faithfully reflect the average service times, queue size, and arrival interval adopted in the real experiment. The automatic instantiation logic was also kept identical to that used in the instrumented code of Pasid-Validator, ensuring that both the model and the experiment respond equivalently to demand variations. The model was executed in stationary mode, with multiple simulations for each AR value. In each case, the MRT was calculated considering the sum of the times in the queues and in the active servers, according to the classic definition of Little's Law, demonstrated in Equation (5). The experimental setup was intentionally designed with two physical machines to provide a controlled and reproducible environment for validation. This setup allowed for precise instrumentation of vehicular networks with request arrivals, queue occupancy, and service times, ensuring statistical comparability with the SPN model. With this configuration, the model was able to simulate the adaptive behavior of the system, allowing a direct comparison with the results observed in the practical experiment, as will be discussed below.

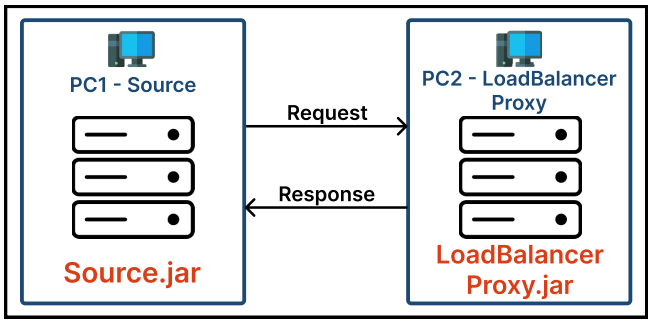


FIGURE 8 | Experimental system architecture. Two networked machines were used: PC1 (Source) generates vehicular requests, while PC2 (Load Balancer) processes them using dynamic autoscaling policies in response to queue occupancy thresholds.

6.3 | Pasid-Validator

The model validation was conducted using the Pasid-Validator tool. This open-source tool was designed to structure the process of comparing SPN performance models, such as the work of [33]. The source code for Pasid-Validator with autoscaling support is publicly available.[2] The tool has several examples of use and was developed to automate the main validation steps. It can be used to collect times in an experimental environment, simulate the model based on this data, perform empirical execution under different load conditions, and perform statistical analysis of the results. In the collection phase, Pasid-Validator instruments the system to capture intermediate times along the path taken by requests, recording points such as the sending of the request, its receipt, and return. These times reflect the real behavior of the distributed system and are used as a basis for parameterizing the timed transitions of the SPN model. This ensures that the simulation operates under realistic conditions that are compatible with the environment. The tool is also responsible for the continuous generation of requests, simulating a constant load flow, using an exponential distribution.

The component responsible for sending keeps track of the AR and records, for each request, the sending time and the response time. At the end of the execution, Pasid-Validator calculates the average response time and its standard deviation based on the collected samples, accurately reflecting the observed dynamic behavior. The comparison between the model results and those obtained in the physical experiment is performed automatically by the tool, using statistical adherence tests, with emphasis on the *t*-test. This process allows for verifying, based on quantitative evidence, whether the simulated values are statistically compatible with the empirical values, within a previously established confidence interval. For this work, because it is open-source, Pasid-Validator was extended with support for autoscaling policies.

The scaling logic has been incorporated directly into the load balancing component: Whenever the occupancy of the request queue exceeds a certain threshold (in this case, 80%), a new service instance is automatically created. This mechanism reflects the policy represented in the SPN model and ensures that the adaptive behavior observed in the simulation is reproduced in the real experiment. With this extension, Pasid-Validator consolidates itself as a tool for validating stochastic models with dynamic characteristics, offering support for both experimental instrumentation and statistical analysis of the results. Its

flexibility and precision allow for validating complex systems under different adaptation strategies, such as reactive scaling and reliability of the results.

6.4 | Validation Results and Comparative Analysis

The comparison between the SPN model's results and the Pasid-Validator data was performed based on the MRT for different values of AR, defined as the inverse of the arrival delay (AD), i.e., AR = 1/AD. This metric represents the rate of arrival of requests per minute and allows the evaluation of the impact of the load intensity on the system's performance. For each AR value analyzed, multiple executions were conducted in the experimental environment, from which the average MRT and the corresponding standard deviation were obtained. The 95% confidence intervals were then calculated based on these samples and used as a reference for the statistical validation of the model results. Table 3 presents the results obtained. It can be observed that, for all load configurations analyzed, the simulated values are within the experimental confidence interval, which indicates statistical adherence between the model and the real system. In addition to statistical consistency, a tendency for MRT to decrease as the AR decreases is observed. This behavior reflects the dynamics expected in systems with adaptive scheduling: With lower load intensity, the system requires fewer instantiations and can operate with fewer active services, maintaining responsiveness within adequate levels.

Figure 9 shows the comparison between experimental and modeled MRT. At the lowest AR (AR = 0.017 msg/m), the experimental MRT was 2049 ms, while the model predicted 2106 ms, a difference of 2.7%. For AR = 0.020 msg/m, the experimental MRT reached 2266 ms, very close to the model value of 2295 ms. In intermediate loads (AR = 0.025 and 0.033 msg/m), the model predicted 3185 and 3359 ms, compared with the experimental 3431 and 3493 ms, remaining within 1.3% deviation. At the highest load (AR = 0.050 msg/m), the experimental MRT was 3599 ms, while the model produced 3795 ms, a maximum difference of 5.4%. These results demonstrate that the SPN model consistently follows the experimental curve across all scenarios, maintaining statistical adherence.

For the lowest load (AR = 0.017 msg/m), the MRT was 2049 ms in the experiment and 2105 ms in the model. As the AR increases, the average response times increase progressively, reaching 3599 and 3794 ms, respectively, for AR = 0.050 msg/m. This evolution reflects the increased pressure on the request queue, requiring more actions from the autoscaling mechanism. The most significant difference between the values, approximately 5.4%, occurs under the highest load and may be the result of startup delays or

TABLE 3 | Comparison between experimental and model results for different arrival rates.

Arrival rate (msg/m)	Exp. MRT (ms)	Std. dev. (exp)	Model MRT (ms)	95% CI (Model)	*t* value	*p*-value
0.017	2049.08	186.28	2105.78	[2071.83; 2139.74]	−0.62	0.540
0.020	2266.25	196.76	2295.20	[2256.23; 2334.17]	−0.48	0.635
0.025	3431.46	311.95	3185.45	[3148.31; 3222.60]	1.33	0.190
0.033	3492.53	317.50	3358.98	[3320.26; 3397.72]	1.12	0.270
0.050	3599.90	327.26	3794.79	[3755.16; 3834.43]	−1.45	0.160

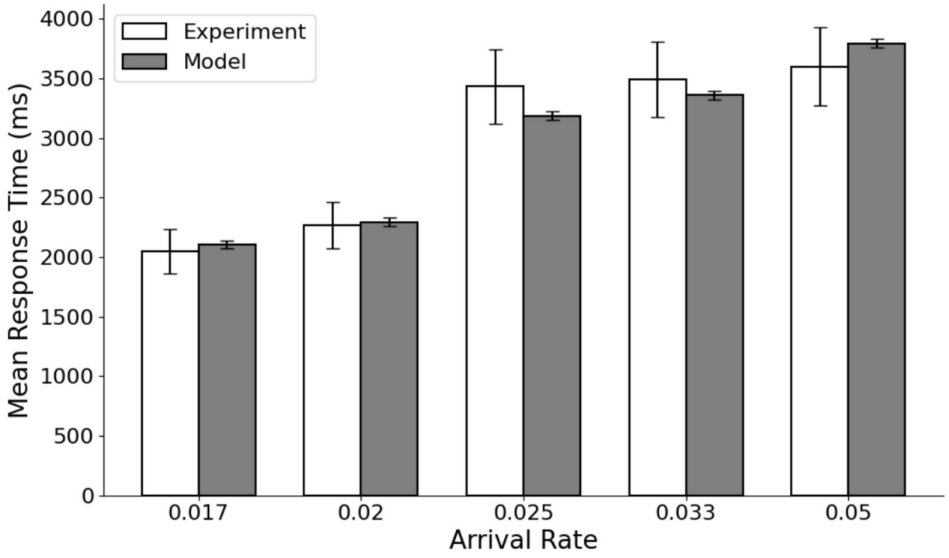


FIGURE 9 | Comparison between experimental and modeled Mean Response Time (MRT) under different arrival rates. The close alignment across all scenarios confirms the statistical validity and accuracy of the SPN model.

transient overloads in the instantiated services. Another aspect observed is the reduction in the variability of the experimental measurements with the decrease in AR.

The most significant error ranges appear in the most loaded scenarios, such as AR = 0.050 and 0.033 msg/m, suggesting greater instability under these conditions. In the intermediate scenarios (AR = 0.025 and 0.020 msg/m), the simulated values closely follow the observed ones, with deviations below 1.3%. In AR = 0.017 msg/m, this difference reduces to 2.7%, demonstrating the accuracy of the model under low-demand conditions.

The results obtained confirm that the proposed SPN model faithfully represents the dynamics of the autoscaling system. In vehicular environments, both the tendency for MRT to increase and its variability under different load levels are observed. The proximity between the simulated and experimental values, with differences lower than 5.4% in all scenarios, validates the adopted approach and demonstrates its potential to support performance analysis. In addition, the model's ability to reflect the stabilization of MRT under light loads and the greater fluctuation under high demand indicates that the instancing mechanisms were correctly incorporated.

Figure 10 shows the system behavior graphs under different load conditions, highlighting the performance of the implemented dynamic autoscaling mechanism. The top panel displays the evolution of the request AR and the number of active services over time. The system is subjected to load variations, with the AR oscillating between different configured values. In response, the number of service instances is dynamically adjusted, alternating between two, three, and four services as demand increases or decreases. This behavior is a direct result of the implementation of the autoscaling logic in Pasid-Validator, which continuously monitors the occupancy of the request queue and instantiates new service threads whenever utilization exceeds the defined threshold.

The bottom panel shows the system utilization over time, calculated from the queue and server occupancy. The red dashed line represents the 80% threshold, which acts as a trigger for reactive scheduling. After a brief initial stabilization phase, the system starts to operate mainly around this threshold, with slight natural oscillations. The eviction trigger is implemented based on ongoing underutilization to avoid a precipitous shutdown of active instances. This mechanism allows the system to react prudently to load drops. This behavior indicates that the scheduling control was successful in keeping utilization close to the point of maximum efficiency, without exceeding the operational limits. The coherence between the variations in the arrival rate, the number of allocated services, and the observed utilization reinforces the fidelity of the implementation in Pasid-Validator, validating the system's adaptive dynamics as represented in the proposed SPN model.

7 | Desing of Experiments

In this section, the DoE approach is used to analyze the influence of different factors on the system energy consumption. DoE provides a systematic methodology to investigate how key variables impact the EC metric, allowing a detailed assessment of EE under different operational scenarios. SPN-based modeling allows a comprehensive representation of the system behavior,

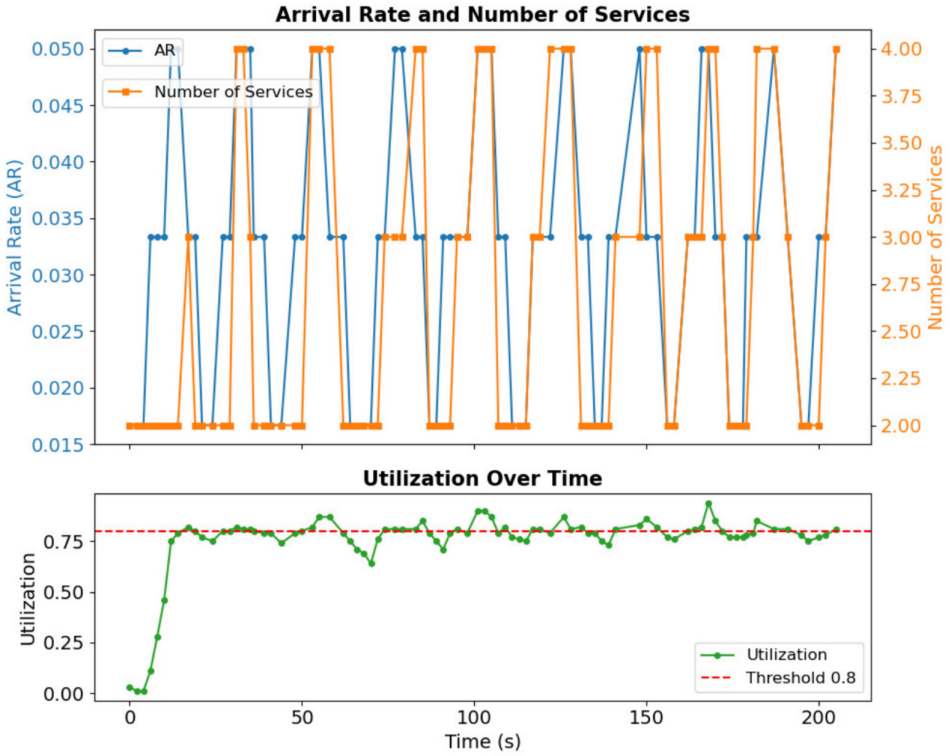


FIGURE 10 | System behavior during dynamic autoscaling. The top panel shows how service instances adjust with varying load. The bottom panel tracks system utilization, with the 80% threshold acting as a trigger for autoscaling, demonstrating adaptive control stability.

considering the dynamics of the components and the variability in the demand for resources [34].

The combination of SPN and DoE allows for the identification of cause-and-effect relationships between system parameters and energy consumption, enabling recommendations for optimization. The experiments conducted in this section explore variations in critical factors, including the number of allocated containers [35], the processing capacity of the containers, and the configuration of the request queues. Statistical analysis of the results allows for the determination of which combinations of parameters result in lower energy consumption, guiding strategies for the more efficient management of computational resources.

7.1 | DoE Overview

DoE is a statistical methodology widely used to plan, conduct, and analyze experiments efficiently. Its goal is to identify the factors that most influence the performance of a system and evaluate their interactions in a controlled manner [36]. In the context of this work, DoE is used to evaluate the impact of different factors on the EE of the container system. It considers variables such as the number of containers, reserved containers, reinstantiation and failure weights, and the size of the request queue.

By applying DoE, we ensure that the variables are manipulated in a way that produces meaningful and reliable results, optimizing the number of experiments required and minimizing computational costs. With this, it is possible to accurately determine how each factor, both individually and in interaction with other factors, affects the EC of the system.

Table 4 presents the factors and levels used in the simulation. The values chosen for the factors and levels were defined based on an analysis of the operational requirements and limitations of the systems under study. The number of containers N_C was varied to reflect the difference between low and high computational capacity scenarios, allowing us to observe the impact of scaling on energy consumption. The variation of C_R aims to test different allocations of reserved containers, which can affect the efficiency and performance of the system. The factors P_R and P_F were adjusted to test different levels of fault tolerance and reinstantiation priorities, with the intention of evaluating how these changes influence the energy consumed. Finally, the size of the queue Q_S, varying, was chosen to simulate different workloads and understand its impact on the EE of the system.

TABLE 4 | Design of experiments.

Factor name	Low setting	High setting
N_C	4.0	8.0
C_R	4.0	48.0
P_R	0.1	0.3
P_F	0.1	0.3
Q_S	100.0	1000.0

Table 5 presents the different combinations of factors and levels tested in the simulation, along with the resulting EC values in kilowatt hours for each configuration. By exploring these combinations, DoE allows identifying which configurations are most energy efficient and provides valuable insights into system behavior under different operating conditions. Applying DoE not only makes it easier to understand the individual effects of each factor but also helps identify complex interactions between factors, providing a more robust and detailed analysis of the

TABLE 5 | Combination of factors considering the energy consumption metric.

N_C	C_R	P_R	P_F	Q_S	EC (kW/h)
4.00	4.00	0.10	0.10	100.00	0.49
4.00	4.00	0.10	0.10	1000.00	0.52
4.00	4.00	0.10	0.29	100.00	0.51
4.00	4.00	0.10	0.29	1000.00	0.51
4.00	4.00	0.29	0.10	100.00	0.57
4.00	4.00	0.29	0.10	1000.00	0.52
4.00	4.00	0.29	0.29	100.00	0.55
4.00	4.00	0.29	0.29	1000.00	0.53
4.00	48.00	0.10	0.10	100.00	0.53
4.00	48.00	0.10	0.10	1000.00	0.60
4.00	48.00	0.10	0.29	100.00	0.51
4.00	48.00	0.10	0.29	1000.00	0.53
4.00	48.00	0.29	0.10	100.00	0.48
4.00	48.00	0.29	0.10	1000.00	0.56
4.00	48.00	0.29	0.29	100.00	0.54
4.00	48.00	0.29	0.29	1000.00	0.47
8.00	4.00	0.10	0.10	100.00	0.54
8.00	4.00	0.10	0.10	1000.00	0.52
8.00	4.00	0.10	0.29	100.00	0.51
8.00	4.00	0.10	0.29	1000.00	0.51
8.00	4.00	0.29	0.10	100.00	0.52
8.00	4.00	0.29	0.10	1000.00	0.50
8.00	4.00	0.29	0.29	100.00	0.51
8.00	4.00	0.29	0.29	1000.00	0.50
8.00	48.00	0.10	0.10	100.00	0.58
8.00	48.00	0.10	0.10	1000.00	0.60
8.00	48.00	0.10	0.29	100.00	0.58
8.00	48.00	0.10	0.29	1000.00	0.52
8.00	48.00	0.29	0.10	100.00	0.51
8.00	48.00	0.29	0.10	1000.00	0.61
8.00	48.00	0.29	0.29	100.00	0.49
8.00	48.00	0.29	0.29	1000.00	0.54

system's energy performance. With this approach, it is possible to optimize system parameters to reduce EC and improve overall efficiency.

Apparent replicates in the table occur because values were rounded to two decimal places due to space constraints in the tabular formatting. In practice, independent replicates of each configuration produced slightly different results, but these differences are not fully visible in the displayed precision. Multiple replicates were calculated for each scenario, and an ANOVA [37] test was conducted to confirm the statistical significance of the estimated effects, ensuring that the reported results are robust despite the visual similarities.

7.2 | Results Analysis

The subsection presents the results obtained from the analysis of the effects of the factors and their interactions on the performance of the investigated system. Figure 5 presents the values of the effects, arranged in decreasing order, allowing the identification of the most significant factors and their respective interactions. This analysis sought to understand which elements of the system exert the greatest influence and how they interact to determine the overall behavior. The results show that the C_R factor (with an approximate value of 0.021) is the most significant, indicating that it is the main determinant of the energy performance of the system. This effect shows that C_R should be treated as an important control variable in any optimization or adjustment attempt (Figure 11).

Next, the P_F factor stands out, with an effect close to 0.019, confirming its substantial relevance and suggesting that it plays a central and complementary role to the effect of C_R. The third largest contribution is given by the interaction C_R×Q_S, whose effect is approximately 0.017. This specific interaction implies that the combined dependency contains complex relationships that can impact the system's energy consumption. On the other hand, at the bottom of the graph, we find the interactions P_R×Q_S, N_C×Q_S, and P_R×P_F, which present effects of approximately 0.001, 0.002, and 0.003, respectively. These values indicate that these interactions have a practically negligible influence on the overall performance of the system. The low value of their effects suggests that they can be discarded in more in-depth analyses or optimization efforts, given that their impact is marginal.

The Figure 12 graphically represents the DOE interactions, offering a clear view of how the dynamicity of autoscaling influences the combinations between factors and their impact on the system's Energy Consumption. Figure 12a shows the interaction between the C_R and the reinstantiation weight (P_R) on EC. It can be seen that, for P_R = 0.1 (blue line), increasing C_R from 4 to 48 results in a reduction in energy consumption, suggesting that a greater availability of reserved containers improves EE when the reinstantiation weight is low. However, for P_R = 0.3 (red line), the behavior is reversed; a greater number of reserved containers leads to an increase in energy consumption. This interaction suggests that reinstantiation has a relevant impact on total EC and that its effect depends on the number of available containers. When the reinstantiation weight is high, there may be a higher cost associated with migrating or restarting containers, increasing EC as C_R increases. With a low reinstantiation weight, a greater number of reserved containers can reduce the need for new allocations, optimizing energy use.

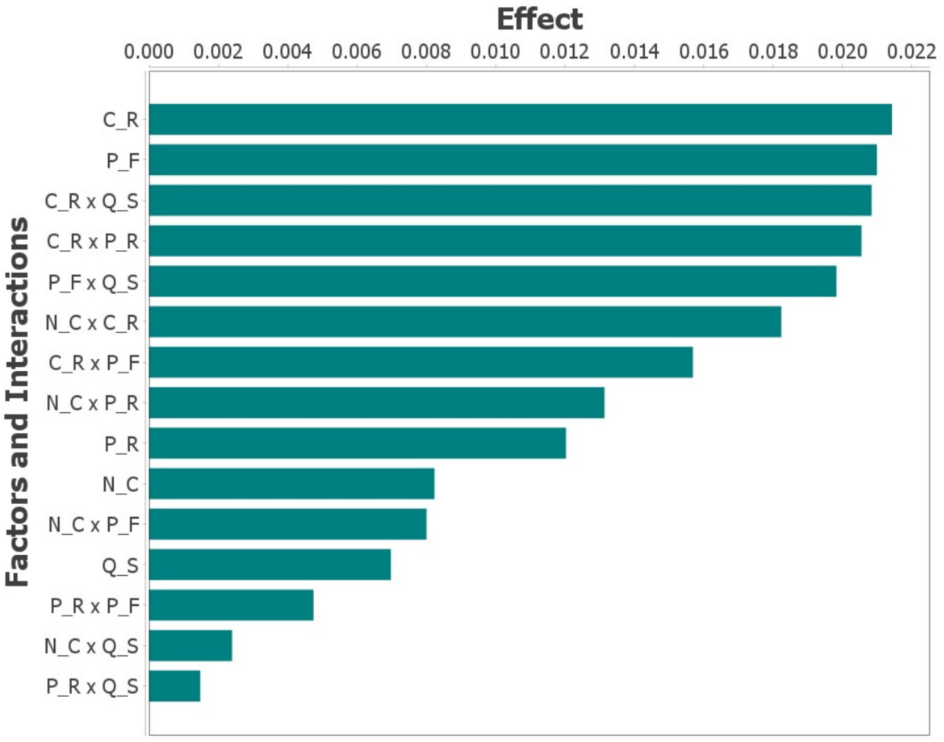
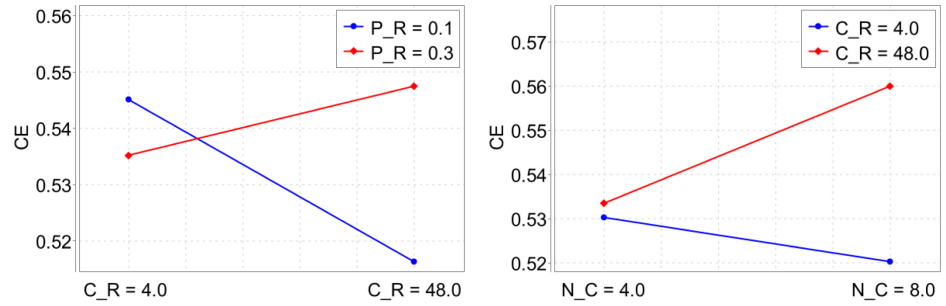
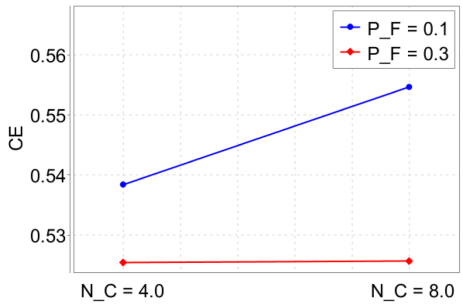


FIGURE 11 | Bar graph showing the effect of different model factors on energy consumption, based on DoE analysis. Reserved containers (C_R) and failure weight (P_F) are identified as the most impactful parameters on energy usage.



(a) Reserved containers x Reinstantiation weight.

(b) Number of containers x Reserved containers.



(c) Number of containers x Fail Weight.

FIGURE 12 | Interaction graphs between key model parameters and their influence on energy consumption: (a) Reserved containers (C_R) vs. reinstantiation weight (P_R); (b) Number of containers (N_C) vs. reserved containers (C_R); (c) Number of containers (N_C) vs. failure weight (P_F).

Next, Figure 12b analyzes the interaction between the N_C and the C_R. It can be seen that, for C_R = 4.0 (blue line), the increase in the total number of containers from 4 to 8 results in a slight reduction in energy consumption, possibly due to a better distribution of the processing load, reducing the impact of overload on individual units. In contrast, for C_R = 48.0 (red line), there is a significant increase in EC as the total number of containers grows. When there are many reserved containers, the addition of new operational instances can increase energy costs due to the need to maintain multiple active units. The difference in the slope of the lines indicates that the relationship between N_C and CE strongly depends on the number of reserved containers, reinforcing the importance of an adequate balance between scalability and EE.

Finally, Figure 12c presents the relationship between the N_C and the P_F on EC. It can be seen that, for P_F = 0.1 (blue line), increasing N_C from 4 to 8 results in a considerable increase in energy consumption, indicating that when the probability of failure is low, adding new containers can generate a greater energy impact due to the cost of maintaining additional instances in continuous operation. In contrast, for P_F = 0.3 (red line), EC remains practically constant, suggesting that a higher failure weight can result in lower container uptime or higher turnover, reducing total energy consumption. This difference in trends suggests that the impact of autoscaling on EC is important, as it reinforces the need to consider dynamism to improve and optimize the infrastructure to minimize energy costs.

It is worth noting that this idealized DoE analysis was delimited with two levels, focusing on the main effects and selected two-factor interactions. Higher-order interactions and nonlinear effects were not explored within the scope of this work. More advanced approaches, such as full factorial designs or Response Surface Methodology (RSM), can provide deeper insights into nonlinear dependencies between parameters and will be considered in future investigations.

8 | Case Studies

This section presents the results obtained as case studies. For the development of the model and execution of the experiments, the Mercury tool [38], version 5.0.1, was used. Due to the complexity of the model, the case studies were conducted through stationary simulations, incorporating an approximate error margin of 2%. The values of the system components were based on previously published academic sources, most notably the studies [8, 9, 39]. The following two subsections highlight the results obtained through the use of the model. The parameters used for the simulations are presented in Table 6.

8.1 | Case Study 1—Comparative Analysis of EC With and Without Autoscaling

This subsection presents the results of the energy impacts of autoscaling on the system's adaptability. One of the main aspects of the work was the prevention of resource waste. Through the autoscaling system, it was possible to reduce EC and minimize the number of idle containers. In addition, the reinstantiation system stands out, as it contributes to resource savings by correcting failures that occurred during the instantiation of containers. Figure 13 presents the results obtained, comparing two system configurations: one with autoscaling disabled (Auto_S

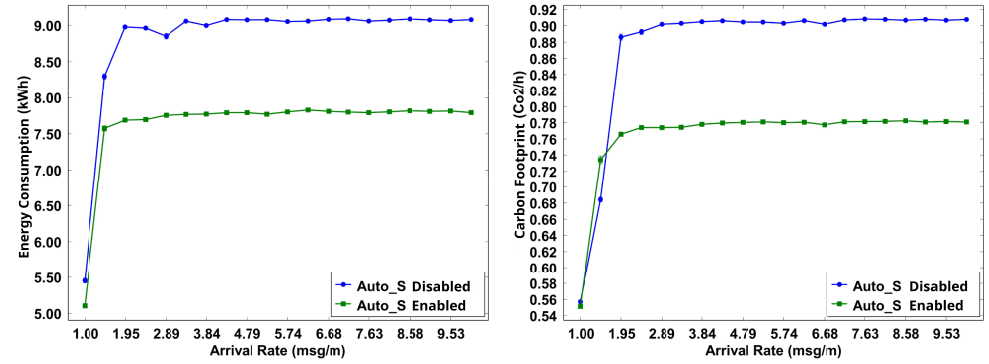
TABLE 6 | Parameters used in the model.

Type	Component	Value	Definition
Places	Pqe	0	Request waiting queue.
	Ppqe	0	Queue of requests to be processed.
	Pcapq	Q_S	Request queue capacity.
	Pprsu	0	RSU processing.
	Pcaprsu	N_C	RSU capacity.
	Pcres	C_R	Reserved containers in the system.
	Pretc	0	Container return place.
	Pic	0	Container instantiation place.
	Plc	0	Container release place.
	Pdf	0	Failure detection place.
	Prc	0	Reinstantiation place.
Timed transitions	Tad	AD	Request arrival time.
	Trsu	0.1	RSU arrival transition.
	Ttsrsu	15.7	Service time transition.
	Tic	1	Container instantiation transition.
	Trc	2	Container reinstantiation transition.
Variables	AD	10	Data arrival.
	Q_S	1000	Queue size.
	N_C	4	Initial number of RSU containers.
	C_R	25	Containers reserved by the system.
	P_F, P_R	0.1, 0.1	Failure and reinstantiation weights.
	E_OP_RSU	0.03	RSU energy consumption.
	E_OP_CONT	0.0001	Container energy consumption.
	Time	60	System operation time.
	CEF	0.01	Carbon emission factor.

Disabled) and another with autoscaling enabled (Auto_S Enabled).

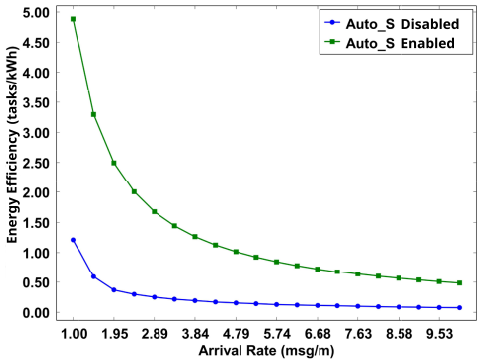
Figure 13a shows the relationship between AR (msg/m) and EC (kWh). At 1.00 msg/m, Auto_S Enabled is approximately 5.1 kWh and Auto_S Disabled approximately 8.3 kWh, a reduction of about 39% with autoscaling. At 1.95 msg/m, consumption rises to approximately 7.7 kWh (enabled) compared to approximately 8.9 kWh (disabled), about 14% lower with autoscaling. At 2.89 msg/m, values are approximately 7.8 kWh (enabled) in contrast to approximately 8.9 kWh (disabled), a reduction close to 12%. For medium-to-high loads (3.84–9.53 msg/m), the non-autoscaling curve stays near 9.0–9.1 kWh, while autoscaling remains around 7.8–8.0 kWh. For example, at 7.63 msg/m it is approximately 7.98 kWh (enabled) compared with approximately 9.06 kWh (disabled), about 12% lower; at 9.53 msg/m, approximately 7.9 kWh relative to approximately 9.05 kWh, about 13% lower. Overall, autoscaling yields consistent savings of 12%–14% under medium loads, with a larger gain (39%) at very low load due to deactivation of idle capacity.

Figure 13b shows the behavior of the system's CF (CO_2/h) as a function of the arrival rate. At 1.00 msg/m, the curves are essentially equal, with Auto_S Enabled approximately 0.56 CO_2/h and Auto_S Disabled approximately 0.54 CO_2/h (a residual difference at very low load). From 1.95 msg/m onward, the advantage of autoscaling becomes evident: approximately 0.75 CO_2/h (enabled) compared to approximately 0.89 CO_2/h (disabled), a reduction of about 16%. At 2.89 msg/m the values are approximately 0.76 (enabled) in contrast to approximately 0.90 (disabled), again about 16% lower with autoscaling. For medium to high loads (3.84–9.53 msg/m), the non-autoscaling curve remains around 0.90–0.92 CO_2/h, while autoscaling stays near 0.78–0.81 CO_2/h. For example, at 7.63 msg/m it is approximately 0.80 (enabled) compared with approximately 0.91 (disabled), about 12% lower; and at 9.53 msg/m, approximately 0.80 relative to approximately 0.91, about 13% lower. In summary, except at



(a) Energy Consumption.

(b) Carbon Footprint.



(c) Energy Efficiency.

FIGURE 13 Energy impact analysis comparing scenarios with autoscaling enabled vs. disabled: (a) Total energy consumption over varying message arrival rates; (b) Resulting carbon footprint (CO_2/h); (c) Energy efficiency measured in tasks per kWh. Autoscaling consistently shows better performance and sustainability.

example, at 7.63 msg/m itis approximately 7.98 kWh (enabled) compared with approximately 9.06 kWh (disabled), about 12% lower; at 9.53 msg/m, approximately 7.9 kWh relative to approximately 9.05 kWh, about 13% lower. Overall, autoscaling yields consistent savings of 12–14% under medium loads, with a larger gain (39%) at very low load due to deactivation ofidle capacity.

Figure 13b shows the behavior ofthe system's Carbon Footprint (CO2/h) as a function ofthe arrival rate. At 1.00 msg/m, the curves are essentially equal, with Auto_S Enabled approximately 0.56 CO2/h and Auto_S Disabled approximately 0.54 CO2/h (a residual difference at very low load). From 1.95 msg/m onward, the advantage of autoscaling becomes evident: approximately 0.75 CO2/h (enabled) compared to approximately 0.89 CO2/h (disabled), a reduction of about 16%. At 2.89 msg/m the values are approximately 0.76 (enabled) in contrastto approximately 0.90 (disabled), again about 16% lower with autoscaling. For medium to high loads (3.84–9.53 msg/m), the non-autoscaling curve remains around 0.90–0.92 CO2/h, while autoscaling stays near 0.78–0.81 CO2/h. For example, at 7.63 msg/m itis approximately 0.80 (enabled) compared with approximately 0.91 (disabled), about 12% lower; and at 9.53 msg/m, approximately 0.80 relative to approximately 0.91, about 13% lower. In summary, except atthe lowest load point

FIGURE 13 | Energy impact analysis comparing scenarios with autoscaling enabled vs. disabled: (a) Total energy consumption over varying message arrival rates; (b) Resulting carbon footprint (CO_2/h); (c) Energy efficiency measured in tasks per kilowatt-hour. Autoscaling consistently shows better performance and sustainability.

the lowest load point (1.00 msg/m), autoscaling consistently reduces the CF by roughly 12%–16% across the evaluated range.

Figure 13c illustrates the behavior of EE (tasks/kWh) as a function of the arrival rate. At 1.00 msg/m, Auto_S Enabled is approximately 4.9 tasks/kWh, while Auto_S Disabled is approximately 1.3 tasks/kWh (about 3.8×higher with autoscaling). As the rate increases to 1.95 msg/m, values are approximately 3.4 (enabled) compared to approximately 0.5 (disabled), about 6.8×higher. At 2.89 msg/m, approximately 2.5 relative to

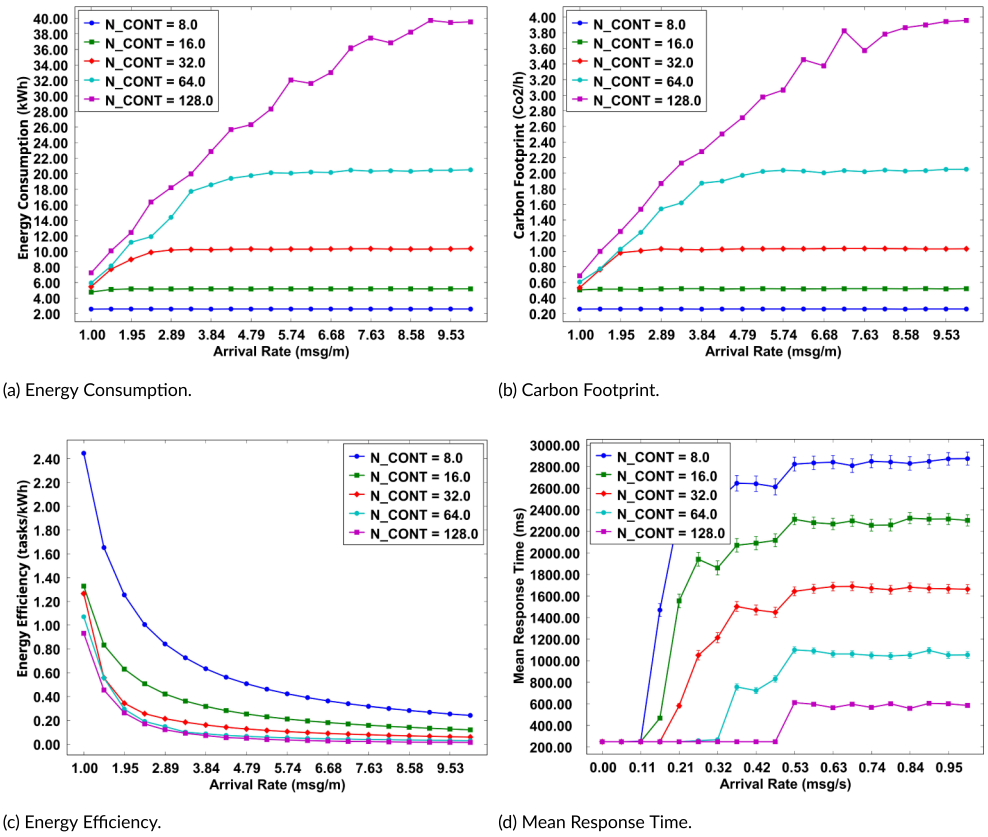
approximately 0.25 (about 10×). At 3.84 msg/m, approximately 1.9 compared with approximately 0.17 (about 11×). At 4.79 msg/m, approximately 1.6 in contrast to approximately 0.12 (about 13×). At 5.74 msg/m, approximately 1.3 compared to approximately 0.10 (about 13×). At 6.68 msg/m, approximately 1.1 relative to approximately 0.08 (about 14×). At 7.63 msg/m, approximately 0.95 compared with approximately 0.07 (about 14×). At 8.58 msg/m, approximately 0.85 in contrast to approximately 0.06 (about 14×). At 9.53 msg/m, approximately 0.78 compared with approximately 0.05 (about 15×). Both curves decrease as load grows, but autoscaling preserves substantially higher efficiency across the entire range.

8.2 | Case Study 2—Influence of the Number of Containers on Energy Performance

In this subsection, the variation in the number of containers in the system is used to analyze the energy performance results. Throughout the analysis, it was observed that the number of containers appears as one of the main factors. Therefore, the autoscaling system was used to vary the number of containers available. Figure 14 presents the results of this variation in the number of containers in search of the best configurations, thus promoting greater sustainability and energy performance of the system.

Figure 14a presents the relationship between EC (in kWh) and message AR (msg/m) for different values of N_CONT, representing the number of containers in the system. The results show that EC increases significantly with the number of containers, especially at higher arrival rates. For N_CONT = 8, consumption is the lowest among the scenarios, remaining constant and below 0.4 kWh, even with high arrival rates. The scenario with N_CONT = 16 presents slightly higher consumption, but stable at around 0.6 kWh, while for N_CONT = 32, consumption increases slightly, stabilizing at approximately 0.8 kWh. In the cases of N_CONT = 64 and N_CONT = 128, a different behavior is observed. With N_CONT = 64, the initial consumption is around 1.0 kWh, gradually increasing until stabilizing around 1.8 kWh. The scenario with N_CONT = 128 records the highest consumption, starting at 1.5 kWh and increasing almost linearly up to 3.5 kWh at the highest rates. The results indicate that the increase in the number of containers is directly associated with higher energy consumption. In scenarios with low demand (reduced msg/m rates), using fewer containers may be more energy efficient. However, in situations of high demand, increased consumption is inevitable due to the need for greater capacity to handle the request load.

Figure 14b examines the CF (in CO_2/h) in relation to the message AR (msg/m) for different amounts of containers. For N_CONT = 8, the CF remains stable around 0.2 CO_2/h,



(a) Energy Consumption.

(b) Carbon Footprint.

(c) Energy Efficiency.

(d) Mean Response Time.

FIGURE 14 | Evaluation of how container quantity affects system performance and sustainability: (a) Energy consumption across message arrival rates; (b) Carbon footprint for each configuration; (c) Energy efficiency; (d) Mean Response Time (MRT). The results illustrate trade-offs between scalability and energy-aware operation.

indicating environmental efficiency with fewer containers. For N_CONT = 16 and N_CONT = 32, the environmental impact increases slightly as the AR increases, ranging from 0.4 to 0.8 CO_2/h. With N_CONT = 64, the ecological implications grow more significantly, exceeding 2 CO_2/h when the AR exceeds 5 msg/m. In the case of N_CONT = 128, a sharp increase is observed, reaching 4 CO_2/h at the highest rates, evidencing an exponential growth in emissions. Although configurations with many containers, such as N_CONT = 128, are suitable for high arrival rates, this comes at the cost of a considerably higher environmental impact. Configurations with fewer containers, such as N_CONT = 8, are more sustainable, but may not meet demand in high-load scenarios. Therefore, it is necessary to balance performance and environmental sustainability.

Figure 14c presents the EE results in tasks/kWh for different container configurations. N_CONT = 8, the efficiency is higher at low arrival rates, starting at 2.4 tasks/kWh and decreasing to less than 0.4 tasks/kWh at rates above 5 msg/m. For N_CONT = 16, the initial efficiency is lower, starting at 1.2 tasks/kWh and stabilizing around 0.4 tasks/kWh at high rates. In the cases of N_CONT = 32, 64, and 128, the initial efficiency values are even lower, dropping quickly to less than 0.2 tasks/kWh at intermediate rates. The results indicate that EE decreases as the number of containers increases, especially in high-load scenarios. Although configurations with few containers, such as N_CONT = 8, are more energy efficient, they may not meet high demands. On the other hand, configurations with many containers, such as N_CONT = 128, are low energy efficient even under high-load conditions, reflecting a less sustainable use of energy.

Figure 14d illustrates the variation of the MRT as the message AR (msg/m) increases, considering different numbers of containers allocated in the system. The observed behavior highlights the direct impact of scalability on the infrastructure's responsiveness. In scenarios with a lower number of containers (N_CONT = 8.0 and 16.0), the MRT grows rapidly after the AR exceeds 2.0 msg/m. For N_CONT = 8.0, for example, the MRT exceeds 2600 ms from 2.89 msg/m, demonstrating an apparent saturation of the system. The same pattern is repeated for N_CONT = 16.0, although with a more damped curve. On the other hand, more robust configurations (N_CONT = 64.0 and 128.0) maintain the MRT at low and stable levels even under high loads. In the case of N_CONT = 128.0, the system maintains the MRT below 500 ms throughout the entire arrival range analyzed, demonstrating high resilience to overload. This demonstrates the effectiveness of the autoscaling mechanism, as it shows that the greater the availability of containers, the greater the system's ability to maintain responsiveness in the face of abrupt variations in demand.

9 | Conclusion

This study presented a model based on SPNs to optimize the dynamic allocation of computational resources in RSUs within VANETs, aiming to reduce EC and CF without compromising system performance. The adopted methodology, combining mathematical modeling with DoE-based experimentation, allowed a rigorous quantitative evaluation of the impacts of autoscaling and the variation in the number of containers on system efficiency. The results demonstrated that autoscaling significantly reduces energy consumption, especially in variable load scenarios, where the dynamic allocation of containers avoids wasting resources. The CF analysis reinforced the proposal's relevance, evidencing a significant reduction in CO_2 without compromising service quality. In addition, a critical relationship was identified between the number of available containers and the system's EE. While leaner configurations are more energy-efficient, scalability must be carefully balanced to ensure responsiveness under high demand.

The proposed model was validated through controlled experiments with the Pasid-Validator tool, which integrated real measurements into the SPN model. The simulated MRT results remained, in all scenarios, within the 95% confidence interval of the experimental data, with *p*-values greater than 0.05 and *t* values close to zero, proving the statistical compatibility between the model and reality. This adherence reinforces the fidelity of the modeling and the model's ability to accurately represent the system's adaptive behavior. The proposed approach stands out for its flexibility in adapting to different traffic patterns, making it a viable alternative for planning innovative and sustainable infrastructures. SPNs have proven effective as a modeling tool, representing complex scenarios and enabling detailed analysis of the impact of operational variables on system performance. It is important to note that this study did not aim to compare the proposed model with other autoscaling strategies, such as threshold-based, predictive, or machine learning-driven approaches. Benchmarking is outside the defined scope, which focuses on validating stochastic MSW modeling with energy-driven approaches. However, as an extension of this work, as a future perspective, we intend to integrate comparisons with state-of-the-art autoscaling strategies and machine-learning techniques to improve decision-making in the autoscaling process, enabling predictive adjustments based on historical traffic patterns and energy consumption. Furthermore, new strategies will be explored to minimize the environmental impact of computational operations in VANETs, seeking solutions that balance sustainability and high performance, especially in critical urban mobility scenarios.

Data Availability Statement

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Endnotes

[1] https://github.com/LuisGuilherme-x7/Pasid_Validator_Autoscaling.

[2] https://github.com/LuisGuilherme-x7/Pasid_Validator_Autoscaling.

References

1. L. G. Silva, I. Cardoso, C. Brito, et al., "Urban Advanced Mobility Dependability: A Model-Based Quantification on Vehicular Ad Hoc Networks With Virtual Machine Migration," *Sensors* 23, no. 23 (2023): 9485.

2. M. Elhoseny and K. Shankar, "Energy Efficient Optimal Routing for Communication in VANETs via Clustering Model," in *Emerging Technologies for Connected Internet of Vehicles and Intelligent Transportation System Networks: Emerging Technologies for Connected and Smart Vehicles* (Springer International Publishing, 2019), 1–14.

3. SERPRO, Registro Nacional de Acidentes e Estatísticas de Trânsito; 2023. Acesso em: 28 nov. 2024, https://www.gov.br/transportes/pt-br/assuntos/transito/arquivos-senatran/docs/renaest.

4. M. Elhoseny, I. M. El-Hasnony, and Z. Tarek, "Intelligent Energy Aware Optimization Protocol for Vehicular Adhoc Networks," *Scientific Reports* 13, no. 1 (2023): 9019.

5. S. Parashar and R. Tiwari, "Traffic Control and QoS Improvement Analysis in V-To-V and V-To-RSU Communication in VANET," in *2023 World Conference on Communication & Computing (WCONF)* (IEEE, 2023), 1–5.

6. M. AlMarshoud, M. Sabir Kiraz, and A. H. Al-Bayatti, "Security, Privacy, and Decentralized Trust Management in VANETs: A Review of Current Research and Future Directions," *ACM Computing Surveys* 56, no. 10 (2024): 1–39.

7. L. Feitosa, P. A. Rego, and F. A. Silva, "Avaliaçao de Desempenho de Migraçao ao Vivo de Contêineres Com Redes de Petri Estocásticas," in *Anais do XXIV Workshop de Testes e Tolerância a Falhas SBC* (SBC, 2023), 94–107.

8. I. Fé, R. Matos, J. Dantas, et al., "Performance-Cost Trade-Off in Auto-Scaling Mechanisms for Cloud Computing," *Sensors* 22, no. 3 (2022): 1221.

9. L. Silva, C. Brito, I. Cardoso, et al., "Desvendando a Elasticidade de Máquinas Virtuais em VANETs: Uma Estratégia Para Aperfeiçoar o Planejamento de Capacidade em RSUs," in *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (SBC, 2024), 169–182. https://sol.sbc.org.br/index.php/sbrc/article/view/29791.

10. F. A. Silva, C. Brito, G. Araújo, et al., "Model-Driven Impact Quantification of Energy Resource Redundancy and Server Rejuvenation on the Dependability of Medical Sensor Networks in Smart Hospitals," *Sensors* 22, no. 4 (2022): 1595.

11. D. Carvalho, L. Rodrigues, P. T. Endo, S. Kosta, and F. A. Silva, "Edge Servers Placement in Mobile Edge Computing Using Stochastic Petri Nets," *International Journal of Computational Science and Engineering* 23, no. 4 (2020): 352–366.

12. P. R. M. Maciel, *Performance, Reliability, and Availability Evaluation of Computational Systems, Volume 2: Reliability, Availability Modeling, Measuring, and Data Analysis* (CRC Press, 2023).

13. S. Sachan, "Stochastic Simulation: An Introduction to Generalized Stochastic Petri Net (GSPN)," Medium 2024, https://medium.com/@swatisachan/stochastic-simulation-an-introduction-generalized-stochastic-petri-net-gspn-dafb72898916.

14. F. Campolongo, S. Tarantola, and A. Saltelli, "Tackling Quantitatively Large Dimensionality Problems," *Computer Physics Communications* 117, no. 1 (1999): 75–85.

15. L. Santos, B. Cunha, I. Fé, M. Vieira, and F. A. Silva, "Data Processing on Edge and Cloud: A Performability Evaluation and Sensitivity Analysis," *Journal of Network and Systems Management* 29, no. 3 (2021): 27.

16. L. Feitosa, G. Gonçalves, T. A. Nguyen, J. W. Lee, and F. A. Silva, "Performance Evaluation of Message Routing Strategies in the Internet of Robotic Things Using the D/M/c/K/FCFS Queuing Network," *Electronics* 10, no. 21 (2021): 2626.

17. I. Costa, J. Araujo, J. Dantas, E. Campos, F. A. Silva, and P. Maciel, "Availability Evaluation and Sensitivity Analysis of a Mobile Backend-As-A-Service Platform," *Quality and Reliability Engineering International* 32, no. 7 (2016): 2191–2205.

18. A. Raza, S. H. R. Bukhari, F. Aadil, and Z. Iqbal, "An UAV-Assisted VANET Architecture for Intelligent Transportation System in Smart Cities," *International Journal of Distributed Sensor Networks* 17, no. 7 (2021): 25–39.

19. N. Rajendran, R. P. Kumar, T. Kavitha, T. Jayakumar, and V. Nanammal, "Delay-Tolerant Load Balancing Routing Model for RSU-Assisted Distributed Vehicular Adhoc Networks," in *2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* (IEEE, 2023), 1–7.

20. N. A. Mohammed, S. M. M. Almutoki, R. Mansoor, A. K. Jaber, B. A. H. K. A. Ghzawi, and A. H. Alsalamy, "Resource Allocation With Energy Balancing for UAVs Assisted VANETs Based Intelligent Transportation System," in *2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT)* (IEEE, 2023), 282–287.

21. T. Mehta and D. P. Mahato, "Effective Scheduling and Nature Inspired Hybrid Load Balancing in VANETs," in *8th International Conference on Computing in Engineering and Technology (ICCET 2023)*, vol. 2023 (IEEE, 2023), 266–273.

22. V. Sethi, S. Pal, and A. Vyas, "Online Energy-Efficient Scheduling Algorithm for Renewable Energy-Powered Roadside Units in VANETs," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (IEEE, 2020), 506–514.

23. Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load Balance Guaranteed Vehicle-To-Vehicle Computation Offloading for Min-Max Fairness in VANETs," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 8 (2022): 11994–12013.

24. S. E. Haider, M. F. Khan, and Y. Saeed, "Adaptive Load Balancing Approach to Mitigate Network Congestion in VANETS," *Compute* 13, no. 8 (2024): 57–69, https://www.mdpi.com/2073-431X/13/8/194.

25. T. Y. Wu, M. S. Obaidat, and H. L. Chan, "QualityScan Scheme for Load Balancing Efficiency in Vehicular Ad Hoc Networks (VANETs)," *Journal of Systems and Software* 104 (2015): 60–68, https://www.sciencedirect.com/science/article/pii/S0164121215000321.

26. R. Qun and S. M. Arefzadeh, "A New Energy-Aware Method for Load Balance Managing in the Fog-Based Vehicular Ad Hoc Networks (VANET) Using a Hybrid Optimization Algorithm," *IET Communications* 15, no. 13 (2021): 1665–1676, https://doi.org/10.1049/cmu2.12179.

27. S. Agarwal, A. Das, and N. Das, "An Efficient Approach for Load Balancing in Vehicular Ad-Hoc Networks," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (IEEE, 2016), 1–6.

28. G. M. N. Ali, M. A. S. Mollah, S. K. Samantha, and S. Mahmud, "An Efficient Cooperative Load Balancing Approach in RSU-Based Vehicular Ad Hoc Networks (Vanets)," in *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)* (IEEE, 2014), 52–57.

29. G. M. N. Ali, E. Chan, and W. Li, "On Scheduling Data Access With Cooperative Load Balancing in Vehicular Ad Hoc Networks (VANETs)," *Journal of Supercomputing* 67 (2014): 438–468.

30. V. Vijayakumar and K. S. Joseph, "Adaptive Load Balancing Schema for Efficient Data Dissemination in Vehicular Ad-Hoc Network VANET," *Alexandria Engineering Journal* 58, no. 4 (2019): 1157–1166.

31. R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling* (John Wiley & Sons, 1990).

32. "Coherence, 10 Best Practices for Effective Auto-Scaling on AWS," 2023, accessed: 2025-05-05, https://www.withcoherence.com/articles/10-best-practices-for-effective-auto-scaling-on-aws.

33. L. Feitosa, V. Barbosa, A. Sabino, et al., "A Comprehensive Performance Evaluation of Container Migration Strategies," *Computing* 107, no. 2 (2025): 1–39.

34. FUKUDA IMea, "Design of Experiments (DoE) Applied to Pharmaceutical and Analytical Quality by Design (QbD)," *Science of the Total Environment* 54 (2018): e01006.

35. D. Lendrem, M. Owen, and S. Godbert, "DOE (Design of Experiments) in Development Chemistry: Potential Obstacles," *Organic Process Research & Development* 5, no. 3 (2001): 324–327.

36. S. A. Weissman and N. G. Anderson, "Design of Experiments (DoE) and Process Optimization. A Review of Recent Publications," *Organic Process Research & Development* 19, no. 11 (2015): 1605–1633.

37. B. K. Das, D. N. Jha, S. K. Sahu, A. K. Yadav, R. K. Raman, and M. Kartikeyan, "Analysis of Variance (ANOVA) and Design of Experiments," in *Concept Building in Fisheries Data Analysis* (Springer, 2022), 119–136.

38. P. Maciel, R. Matos, B. Silva, et al., "Mercury: Performance and Dependability Evaluation of Systems With Exponential, Expolynomial, and General Distributions," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)* (IEEE, 2017), 50–57.

39. "Gov, Geração de Energia Elétrica Tem a Menor Emissão de CO_2 Dos Últimos Onze Anos," 2024, acesso em: 29 nov. 2024, https://agenciagov.ebc.com.br/noticias/202402/geracao-de-energia-eletrica-tem-a-menor-emissao-de-co2-dos-ultimos-onze-anos-1.