IoT-TSim: A Lightweight Discrete-Event Simulator for Gateway Queueing and Processing Dynamics in IoT

Melissa Alves[1], Vandirleya Barbosa[1], Eduardo Cerqueira[2], Denis Rosário[2], Iure Fé[1], Ermeson Andrade[3], Francisco Airton Silva[1]

[1]Universidade Federal do Piauí (UFPI)

[2]Universidade Federal do Pará (UFPA)

[3]Universidade Federal Rural de Pernambuco (UFRPE)

{melissaalves,vandirleya.barbosa,iure.fe, faps}@ufpi.edu.br

{cerqueira,denis}@ufpa.br

ermeson.andrade@ufrpe.br

Abstract. *Internet of Things (IoT) architectures with many devices sending data to an edge gateway are common in monitoring and telemetry applications, where many design decisions depend primarily on the temporal behavior of message flows rather than on physical constraints. This paper presents a lightweight, time-oriented discrete-event simulator that models IoT devices as stochastic message sources and the gateway as a finite-capacity system with parallel processing. Simulation results show that increasing the number of devices accelerates gateway saturation, leading to sharp increases in end-to-end delay and packet loss. In contrast, increased gateway parallelism delays saturation and raises the maximum achievable throughput, supporting quantitative gateway dimensioning and traffic planning.*

1. Introduction

Recent years have witnessed a rapid expansion of Internet of Things (IoT) technologies, with new devices introduced to the market almost daily [Dauda et al. 2023]. This is due to IoT becoming one of the most disruptive technologies in recent years, where the number of connected devices is growing at an unprecedented pace. According to recent global estimates, the number of IoT connections worldwide is expected to exceed 40 billion by 2034 [Statista Research Department 2026]. This growth reflects the widespread adoption of IoT technologies across different domains (i.e., consumer and industrial), as IoT is increasingly embedded in real-world systems that generate and exchange massive volumes of data [Elgazzar et al. 2022]. Hence, this expansion underscores the importance of scalable and efficient data-processing architectures.

In this context, it is evident that as the IoT market expands, attracting an increasing attention from both academia and industry [Nawab and Shadmon 2024]. This growth results in environments characterized by numerous heterogeneous devices continuously generating data, which imposes challenges on performance evaluation and bottleneck identification, particularly at the gateway. This is because the gateway serves as the central point for aggregating and processing IoT messages. Therefore, understanding the temporal behavior of traffic generated by IoT devices and its impact on the processing infrastructure is essential to ensure the quality and reliability of these applications.

In this scenario, the concentration of thousands or even millions of IoT devices transmitting data to a single gateway represents a significant challenge in terms of performance and scalability, since the high dispersion and large number of devices can hinder the prediction of system performance [Araujo et al. 2025]. Message overload may lead to queue growth, increased latency, and even data loss, especially when generation patterns are stochastic or subject to traffic bursts. An alternative to waiting for physical devices to be developed and built is using IoT simulators to test and evaluate prospective IoT devices, systems, and designs [Dayalan et al. 2022]. However, many existing approaches rely heavily on detailed modeling of the physical layer, network topology, or specific protocols, which hinders direct assessment of the impact of message temporal behavior on the gateway and limits agile exploration of purely temporal scenarios.

Several IoT simulators have been proposed to support performance evaluation under different assumptions, including tools focused on resource allocation and service orchestration across Edge, Fog, and Cloud layers [Cantero et al. 2025, Mahmud et al. 2022, Zeng et al. 2017, Mechalikh et al. 2021] and solutions that incorporate mobility, radio technologies, or network topology [Firouzabadi et al. 2023, Lera et al. 2019]. While these approaches offer detailed modeling capabilities, they often rely on physical-layer details, complex infrastructures, or external simulation engines, which limit the agile exploration of purely temporal parameters. Therefore, there is still a need for simulation approaches that prioritize gateway behavior and temporal dynamics, abstracting physical-layer aspects while enabling the evaluation of performance metrics under consistent temporal assumptions.

This paper presents a lightweight IoT simulator oriented toward time-based events (IoT-TSim[1]), designed to analyze the impact of IoT traffic from the gateway perspective, where aggregation and processing bottlenecks are concentrated. Rather than providing detailed physical-layer or protocol-level modeling, IoT-TSim deliberately adopts a higher abstraction level focused on temporal message dynamics and gateway behavior. This design enables a direct investigation of congestion, latency, throughput (TP), and packet loss under different load conditions, while also supporting cross-validation with formal analytical models, such as SPNs, under the same temporal assumptions. Simulation results show that IoT-TSim effectively captures gateway saturation effects, including end-to-end delay growth, throughput saturation, and packet loss under increasing load, consistently with the behavior predicted by the SPN model.

The main contributions of this work are the following: *(i)* the proposal of a strictly time-oriented and gateway-centric IoT simulator, suitable for performance and scalability analysis; and *(ii)* the design of IoT devices as stochastic sources of temporal events, enabling a quantitative investigation of the impact of message generation rates, device population, and gateway processing capacity on system behavior, while also supporting gateway dimensioning, capacity planning decisions, and the validation of formal analytical models through consistent comparisons with SPN-based models.

The remainder of this paper is organized as follows: Section 2 presents the related work; Section 3 describes the problem addressed in this work; Section 4 presents the proposed simulator; Section 5 discusses the simulator validation process; Section 6 presents the case studies and analyzes the obtained results; and finally, Section 7 presents the conclusions and directions for future work.

[1]IoT-TSim: https://gitlab.com/pasid3/iot-tsim

2. Related Work

Table 1 summarizes the main characteristics of the analyzed tools, enabling a systematic comparison of their modeling focuses and abstraction levels. Among these works, simulators such as SimulateIoT-Services [Cantero et al. 2025], iFogSim2 [Mahmud et al. 2022], IOTSim [Zeng et al. 2017], and PureEdgeSim [Mechalikh et al. 2021] stand out, forming a group of solutions aimed at resource allocation and microservice management across multiple layers, such as Edge, Fog, and Cloud. Although robust, these tools require detailed modeling of the distributed computing infrastructure and are generally oriented toward Big Data scenarios or service orchestration. In this sense, this issue imposes high computational overhead and increased configuration complexity for analyses focused on the temporal behavior of messages.

Table 1. Comparison of related works.

Ref	Simulator	Gateway-Centric	Physical Layer Abstraction	Validation Method	Time-Based Events	External Independence
[Alavi Fazel and Wainer 2024]	No Name	✓	×	×	×	×
[Cantero et al. 2025]	SimulateIoT	✓	×	×	×	×
[Mahmud et al. 2022]	iFogSim2	✓	×	×	×	×
[Firouzabadi et al. 2023]	PIoT	✓	×	×	×	×
[Zeng et al. 2017]	IoTSim	✓	×	×	×	×
[Ta et al. 2019]	LoRa-MAB	✓	×	×	×	✓
[Bounceur et al. 2018]	CupCarbon	✓	×	×	×	✓
[Sonmez et al. 2018]	EdgeCloudSim	✓	×	✓	✓	×
[Lera et al. 2019]	YAFS	✓	✓	×	✓	✓
[Mechalikh et al. 2021]	PureEdgeSim	✓	✓	×	✓	×
This Work	**IoT-TSim**	✓	✓	✓	✓	✓

A second group of works focuses on large-scale simulators and scenarios that incorporate mobility, as observed in PIoT [Firouzabadi et al. 2023], YAFS [Lera et al. 2019], and PureEdgeSim [Mechalikh et al. 2021]. PIoT stands out for its ability to simulate millions of devices over 4G/5G cellular infrastructures, while YAFS employs complex network theory to model mobility and dynamic failures. However, these simulators maintain an intrinsic dependency on the physical layer and network topology for result generation, which may hinder the agile exploration of variations in stochastic and temporal parameters, especially in scenarios where radio details or geographical location can be abstracted.

In a distinct approach, there are tools grounded in formal specifications and detailed modeling of specific protocols. For instance, [Alavi Fazel and Wainer 2024] employed the Discrete Event System Specification (DEVS) to ensure robustness in residential automation applications, while CupCarbon-Lab [Bounceur et al. 2018] acts as an emulator that enables code injection into real hardware, such as Raspberry Pi devices. EdgeCloudSim [Sonmez et al. 2018] focuses on performance evaluation in Edge Computing systems. LoRa-MAB [Ta et al. 2019] focuses on the radio layer and specific protocols, such as LoRaWAN, addressing aspects of collisions and resource allocation. Despite their high accuracy, these approaches tend to be tightly coupled to external simulation engines or physical protocol details, limiting their portability and focusing primarily on purely temporal analytical analyses.

Unlike the studies discussed in this section, this work does not aim to replace or compete with comprehensive multi-layer IoT simulators that model detailed physical, network, and infrastructure aspects. Instead, it addresses a different problem space by

proposing a lightweight, gateway-centric approach for analyzing temporal message dynamics at a controlled abstraction level. By representing sensors as stochastic sources of temporal events and treating the gateway as the main aggregation and processing point, the simulator enables a direct investigation of congestion, saturation, scalability, and message loss. In addition, its self-contained design and alignment with SPN-based analytical assumptions make it suitable not only for simulation-based performance analysis but also for consistency assessment with formal models. Therefore, the main distinction of this work lies in offering a focused, low-overhead, and analytically compatible approach for temporal performance and scalability studies in IoT systems.

3. Problem Statement

Figure 1 illustrates the architecture of the IoT system considered in this work, highlighting the communication flow between the distributed IoT devices and the central gateway that processes messages. The model represents a basic data collection and processing scenario that does not account for communication failures, physical-layer interference, or adaptive device behavior, and focuses exclusively on the operational cycle of message generation, transmission, and processing.

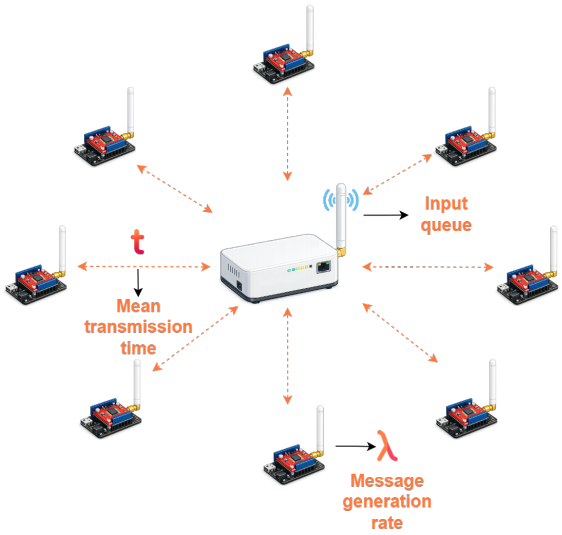


Figure 1. Architecture of the IoT gateway-based system, highlighting message generation at distributed sensors (λ), transmission delay (t), the input queue at the gateway, and centralized message processing.

The architecture consists of a set of IoT devices distributed throughout the environment, as expected in consumer and industrial application domains [Elgazzar et al. 2022]. The IoT devices independently generate messages at an average rate λ, which are transmitted to the gateway through wireless communication links and are subject to a propagation delay associated with the transmission medium. The operational cycle begins at the sensors, which remain active and periodically generate messages. Each generated message is immediately transmitted to the gateway, remaining in transit until it reaches the processing point. Upon arrival at the gateway, the message is forwarded to a finite-capacity waiting queue, which temporarily stores messages awaiting service.

The gateway is equipped with a limited number of parallel-processing cores, which represent its computational capacity. The service begins immediately as soon as

the processing cores are available at the time a message arrives. Otherwise, the message remains in the queue until a core becomes available. If the queue is full upon arrival, the message is discarded, resulting in a loss in the system. During processing, each message exclusively occupies a gateway core for a given service time, after which processing is completed and the message leaves the system. The core is then released and becomes available to serve subsequent messages. For clarity and focus, the model concentrates on this standard message flow, emphasizing the main states and transitions involved in the communication and processing process.

4. IoT-TSim

IoT-TSim is a configurable discrete-event simulation tool for analyzing message traffic patterns in IoT gateway architectures. The system models the complete message lifecycle, from sensor generation to gateway processing, including inter-arrival sampling, propagation delays, queueing, parallel processing, and statistical collection across multiple operational scenarios. Users can define an arbitrary number of IoT devices with individual generation rates, configure gateway processing capacity through parallel servers, and adjust system parameters such as queue capacity and service times. The simulator supports exponential distributions for both inter-arrival times and service durations, with configurable mean values and optional bounds. This design allows practitioners to explore different IoT deployment scenarios without code modifications, enabling comprehensive performance analysis under diverse operational conditions.

The IoT-TSim implementation follows a discrete-event simulation paradigm, structured around a centralized simulation engine, a statistical evaluation framework, and a flexible configuration mechanism. The simulation engine maintains a Future Event List (FEL) implemented as a priority queue, ensuring chronological processing of events throughout the execution. Three fundamental event types are supported: sensor message generation, message arrival at the gateway, and message departure after processing. This event-driven structure enables precise modeling of system dynamics and efficient handling of message flows under different operating conditions. Additionally, IoT-TSim provides a configuration management layer that allows system parameters to be modified without requiring changes to the source code, supporting systematic scenario analysis.

IoT-TSim adopts a deliberate abstraction level focused on temporal message behavior and gateway processing dynamics. Accordingly, the current version of the simulator does not model physical-layer interference, communication failures, retransmissions, protocol-specific effects, or adaptive device behavior. This design choice aligns with the objective of providing a lightweight, reproducible environment for temporal performance analysis, scalability assessment, and gateway dimensioning, rather than detailed protocol- or network-level evaluation.

4.1. Execution Algorithm

The simulation engine employs a single-threaded, event-driven approach in which the main simulation loop processes events in chronological order until the specified simulation time is reached. The core event dispatcher handles each event type through specialized procedures: sensor generation events create new messages and schedule both gateway arrivals (after propagation delay) and next sensor generations; gateway arrival events attempt immediate processing if servers are available, otherwise queue messages

or drop them if capacity is exceeded; departure events free servers, update performance metrics, and initiate processing of queued messages if available.

Algorithm 1 details the main execution flow of IoT-TSim following a discrete-event simulation paradigm. Lines 1–6 initialize the simulation by setting the initial simulation time and scheduling the first message-generation event for each sensor based on its sampled inter-arrival time. Lines 8–18 implement the main simulation loop, which repeatedly extracts the next event from the FEL, advances the simulation clock, and dispatches the event to the appropriate handler based on its type. The procedure `HandleSensorGenerate` (lines 20–26) models message creation at the sensor level, scheduling the gateway's arrival after a propagation delay and the next sensor generation event, if the simulation horizon has not been reached. Gateway arrivals are processed in lines 28–36, where messages are either immediately served if processing resources are available, enqueued if buffer capacity allows, or dropped otherwise. Finally, lines 38–44 describe the gateway departure logic, which releases a processing server, updates performance metrics, and starts service for the next queued message if available.

Algorithm 1 IoT-TSim Main Simulation Loop

```
1: procedure RUN
2:     currentTime ← 0
3:     for all sensor s in sensors do
4:         t ← s.SAMPLEINTERARRIVAL()
5:         SCHEDULE(SENSORGENERATE, t, s)
6:     end for
7:     while FEL not empty and currentTime ≤ simulationTime do
8:         e ← FEL.POLL()
9:         currentTime ← e.time
10:        if e.type = SENSORGENERATE then
11:            HANDLESENSORGENERATE(e)
12:        else if e.type = GATEWAYARRIVAL then
13:            HANDLEGATEWAYARRIVAL(e)
14:        else if e.type = GATEWAYDEPARTURE then
15:            HANDLEGATEWAYDEPARTURE(e)
16:        end if
17:    end while
18: end procedure
19: procedure HANDLESENSORGENERATE(event e)
20:    msg ← CREATEMESSAGE(e.sensor)
21:    delay ← e.sensor.SAMPLEPROPAGATIONDELAY()
22:    SCHEDULE(GATEWAYARRIVAL, currentTime + delay, msg)
23:    next ← e.sensor.SAMPLEINTERARRIVAL()
24:    if currentTime + next ≤ simulationTime then
25:        SCHEDULE(SENSORGENERATE, currentTime + next, e.sensor)
26:    end if
27: end procedure
28: procedure HANDLEGATEWAYARRIVAL(event e)
29:    if busyServers < parallelism then
30:        STARTSERVICE(e.msg)
31:    else if queue.SIZE() < queueCapacity then
32:        queue.ADD(e.msg)
33:    else
34:        droppedCount ← droppedCount + 1
35:    end if
36: end procedure
37: procedure HANDLEGATEWAYDEPARTURE(event e)
38:    busyServers ← busyServers - 1
39:    UPDATEMETRICS(e.msg)
40:    if queue not empty then
41:        nextMsg ← queue.REMOVE()
42:        STARTSERVICE(nextMsg)
43:    end if
44: end procedure
```

4.2. Metrics

The IoT-TSim collects three primary performance metrics during simulation execution, which are automatically calculated and reported via the `getAverageResponseTime()`, `getThroughput()`, and `getDroppedMessages()` methods. The Mean Response Time (MRT) corresponds to the interval between message generation at a sensor and processing completion at the gateway (Eq. 1), encompassing propagation delays, queueing time, and service time. The simulator accumulates the total response time for all processed messages and calculates the mean at the end of each simulation run. This enables the analysis of how sensor generation rates, gateway parallelism, and queue capacity affect overall system performance. The $t_i^{\text{generation}}$ denotes the creation time of message i at the sensor, and $t_i^{\text{completion}}$ corresponds to the processing completion time at the gateway, with N representing the total number of successfully processed messages. It is important to note that the MRT calculation excludes messages that are dropped due to queue saturation, considering only those that complete the entire processing cycle.

$$\text{MRT} = \frac{1}{N} \sum_{i=1}^{N} \left(t_i^{\text{completion}} - t_i^{\text{generation}} \right) \tag{1}$$

The TP measures the system's effective processing rate, defined as the ratio of successfully processed messages to the total simulation time (Eq. 2). This metric reflects the gateway's actual message-processing capacity under different operational conditions and enables the identification of system saturation points. In the Equation, N represents the total number of completed message processings, and T_{sim} denotes the total simulation time duration.

$$\text{TP} = \frac{N}{T_{\text{sim}}} \tag{2}$$

The Dropped Messages (DM) represents the absolute count of messages that are discarded due to system saturation at the gateway. A message is dropped when it arrives at the gateway and finds all processing servers busy and the queue at full capacity. In the Equation 3, G represents the total number of generated messages, m_i denotes message i, and $\mathbf{1}_{\text{drop}}(m_i)$ is an indicator function that equals 1 if message m_i is dropped due to queue saturation, and 0 otherwise. This metric provides an absolute count of system losses, enabling analysis of capacity limits under different load conditions.

$$\text{DM} = \sum_{i=1}^{G} \mathbf{1}_{\text{drop}}(m_i) \tag{3}$$

5. Validation

To validate the behavior observed in IoT-TSim, an analytical model based on SPN was developed using the Mercury tool [Maciel et al. 2017]. This validation approach captures the same message flow implemented in the simulator, enabling a direct comparison between simulated outcomes and theoretical analytical results. By ensuring that timed transitions follow exponential distributions consistent with the assumptions of the discrete-

event simulation, the model provides a consistent baseline for verifying the simulator's logic under identical stochastic hypotheses.

Figure 2 illustrates the SPN model, which is divided into two main logical components: the *Transmission* subsystem and the *Gateway* subsystem. The transmission part models message generation by IoT devices and their propagation toward the gateway, while the gateway part models queueing and processing under limited resource availability. All timed transitions operate in *infinite server* mode, meaning that an unbounded number of tokens can fire concurrently without contention at the transition itself, as defined in classical SPN modeling [Maciel 2023]. System capacity constraints are explicitly enforced via control places that regulate buffer sizes and processing parallelism.

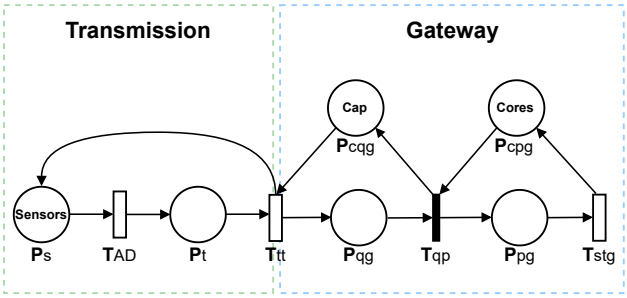


Figure 2. SPN model highlighting places, transitions, and control tokens.

The description of the components is presented in Table 2. The system begins with IoT sensors generating messages via the *Arrival Delay* (*TAD*) transition, which models message generation as an exponential distribution. The place *Ps* represents the aggregated source of IoT events, while *Pt* holds messages ready for transmission. The *Transmission time* (*Ttt*) transition models the propagation delay from sensors to the gateway. At the gateway, *Pqg* represents the input queue with finite capacity controlled by the *Cap* marking place, while *Pcqg* and *Pcpg* manage the gateway's queueing and processing capacity, respectively. The *Service time* (*Tstg*) transition models message processing at the gateway, with available processing cores controlled by the *Cores* marking place.

Table 2. Components of the SPN model.

Type	Component	Description
Timed transitions	TAD	Arrival delay modeling message generation per sensor
	Ttt	Propagation time from sensors to the gateway.
	Tstg	Service time for message processing at the gateway.
Places	Ps	IoT message generation event source.
	Pt	Messages ready for transmission to the gateway.
	Pqg	Gateway queue capacity.
	Pcqg	Gateway input queue.
	Ppg	Gateway processing capacity.
	Pcpg	Messages currently under processing at the gateway.
Marking places	Sensors	Number of IoT sensors generating messages in the system.
	Cap	Maximum capacity of the gateway queue.
	Cores	Number of available gateway processing cores.

5.1. Statistical Comparison

The proposed SPN model was evaluated through a consistency analysis by comparing its analytical results with those from the IoT-TSim simulator, both using the same configuration summarized in Table 3. We analyzed the TP metric to assess agreement between the simulation and analytical results, computing the mean TP and standard deviation for each

approach in each scenario. The statistical comparison employed 95% confidence intervals and the Student's t-test to evaluate the significance of the observed differences.

Table 3. System configuration parameters adopted as the setup for the validation experiments.

Component	Value
Number of sensors	200, 300, 400
Number of gateway cores	2
Queue capacity	50 packets
Mean service time	0.3 s
Mean propagation delay	0.5 s

Table 4 summarizes the validation results. For all analyzed scenarios, the confidence intervals for the TP differences include zero, and the corresponding p-values remain above the 0.05 significance level. This indicates that there are no statistically significant differences between the TP values produced by the simulation and those predicted by the analytical model. Therefore, the close agreement between the SPN predictions and the simulation results validates the proposed model, demonstrating its ability to accurately capture the system's TP behavior across different sensor densities.

Table 4. Comparative validation of throughput between the simulation and the analytical model for different numbers of sensors.

Sensors	TP_Sim	StDev_Sim	TP_Mod	StDev_Mod	95% CI	t-stat	p-value
200	5.775	1.669	5.719	1.698	$[-0.0120, 0.1238]$	1.90	0.094
300	6.185	1.215	6.188	1.184	$[-0.0275, 0.0210]$	-0.31	0.764
400	6.369	0.890	6.361	0.874	$[-0.01140, 0.02847]$	0.99	0.352

Figure 3 provides a visual comparison between the IoT-TSim simulator and the SPN model results for TP under different numbers of IoT devices. The close alignment between the simulated and analytical curves across all evaluated generation rates confirms the consistency observed in the statistical analysis. For all sensor configurations, the model closely matches the simulation results across the evaluated range, with no systematic deviations or trend mismatches. This visual agreement reinforces the results reported in Table 4, where the confidence intervals include zero and the p-values indicate no statistically significant differences between the two approaches.

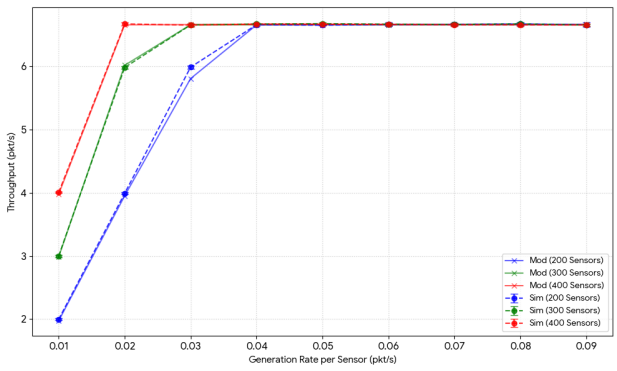


Figure 3. Comparison between analytical model and simulation results for throughput across different sensor configurations.

6. Case Study

This section presents the results obtained from simulations conducted with the proposed IoT-TSim. The objective is to analyze the behavior of a centralized gateway architecture under different traffic and capacity conditions, focusing on key performance metrics such as mean end-to-end time, number of dropped messages, and gateway throughput.

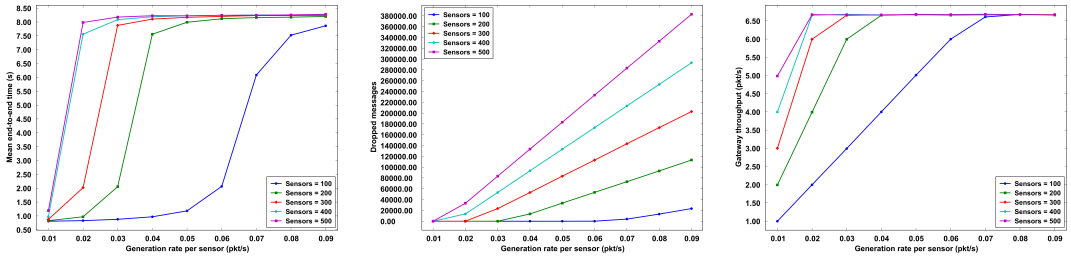
Two scenarios are investigated to assess the impact of workload intensity and service capacity on system performance. In the first scenario, the effect of varying the traffic load generated by IoT devices toward a centralized gateway is analyzed, whereas in the second scenario, the impact of gateway processing capacity is evaluated. The specific parameter values and variations adopted in each scenario are summarized in Table 5. Together, these scenarios allow a systematic characterization of gateway behavior as packet generation rates increase. Each point in both scenarios was executed with 10 replications, using a fixed simulation horizon of 10,000 seconds. No warm-up period was adopted, and a fixed master seed (12345) was used to derive distinct pseudo-random streams for each replication, ensuring reproducible results while preserving variability across runs.

Table 5. System configuration parameters used in the case study experiments.

Parameter	Scenario 1	Scenario 2
Number of sensors	100, 200, 300, 400, 500	100
Number of gateway cores	2	1, 2, 3, 4, 5
Queue capacity	50 packets	50 packets
Mean service time	0.3 s	0.3 s
Mean propagation delay	0.5 s	0.5 s

6.1. Scenario 1: Varying the Number of Sensors

This case study evaluates the impact of increasing the number of sensors on system performance. Figure 4 presents the interaction between the packet generation rate per sensor and the number of sensors for three key performance metrics (mean end-to-end time, number of dropped messages, and gateway throughput). As the generation rate increases, the total arrival rate at the gateway grows proportionally with the number of active IoT devices, leading the system from a stable to a congested operating regime.



(a) Mean End-to-End Time **(b) Dropped Messages** **(c) Throughput**

Figure 4. Impact of the packet generation rate per sensor and the number of sensors on mean end-to-end time, dropped messages, and throughput.

Figure 4a shows the mean end-to-end time as a function of the packet generation rate per sensor. For the scenario with 100 IoT devices, the mean end-to-end time remains

relatively low at small generation rates, increasing gradually until approximately 0.06 packets per second per sensor. Beyond this point, the delay rises sharply, reaching values close to 8 seconds, indicating that the gateway can no longer process incoming messages at the same pace they arrive.

As the number of IoT devices increases, this transition occurs at significantly lower generation rates. For 200 IoT devices, a steep increase in delay is already observed around 0.03 packets per second per sensor. In comparison, for 300 or more IoT devices, the delay escalates at even lower rates, quickly stabilizing near the maximum observed value. This behavior indicates that higher sensor populations reduce the sustainable generation rate per sensor, as the aggregated load drives the gateway into saturation earlier.

Figure 4b depicts the number of dropped messages under the same conditions. For all configurations, the number of dropped messages increases monotonically with the generation rate per IoT device, with steeper growth as the number of sensors increases. While packet drops remain relatively limited for 100 IoT devices at low generation rates, configurations with 300, 400, and 500 IoT devices exhibit a substantial, nearly linear increase in dropped messages across the evaluated range, reflecting sustained overload at the gateway.

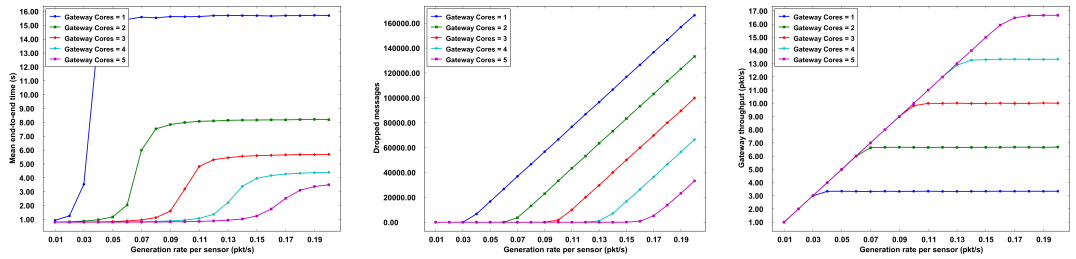
Figure 4c presents the gateway throughput as a function of the packet generation rate per sensor. Initially, throughput increases with the offered load, indicating effective utilization of processing capacity. However, across all sensor configurations, throughput eventually reaches a plateau of around 6.5 packets per second, representing the gateway's maximum processing capacity under the configured service parameters. Scenarios with a larger number of IoT devices reach this throughput ceiling at lower generation rates per sensor; while the configuration with 100 IoT devices requires higher per-device rates to fully saturate the gateway, configurations with 400 and 500 IoT devices achieve maximum throughput almost immediately. Beyond saturation, further increases in generation rate do not improve throughput; instead, they lead to higher delays and increased message losses.

This case study shows that increasing the number of IoT devices significantly affects system scalability, accelerating the onset of congestion. Although gateway throughput is bounded by its service capacity, higher sensor densities reduce the admissible generation rate per sensor, leading to increased queueing delays and packet losses. These results highlight a clear trade-off between sensor density and sustainable traffic generation in gateway-centric IoT architectures.

6.2. Scenario 2: Varying the Number of Gateway Cores

This case study investigates the impact of the gateway processing capacity on system performance by varying the number of available processing cores. Figure 5 illustrates the interaction between the packet generation rate per IoT device and the number of gateway cores for three performance metrics: mean end-to-end time, number of dropped messages, and gateway throughput. As the offered load increases, the system transitions from a stable to a congested operating regime. However, unlike the previous scenario, this transition is strongly influenced by the gateway's processing capacity, which directly determines the amount of traffic it can absorb before saturation.

Figure 5a shows the mean end-to-end time as a function of the packet generation rate per IoT device. With a single processing core, the mean delay increases sharply by



(a) Mean End-to-End Time **(b) Dropped Messages** **(c) Throughput**

Figure 5. Impact of the packet generation rate per sensor and the number of gateway cores on mean end-to-end time, dropped messages, and throughput.

around 0.03 packets per second per device. It quickly stabilizes at high values, around 15 seconds, indicating that the gateway rapidly becomes a bottleneck due to limited processing capacity. As the number of gateway cores increases, this sharp delay growth is progressively shifted to higher generation rates. For two cores, a similar increase is observed around 0.06 packets per second per sensor, while for three, four, and five cores, the transition occurs at even higher rates. At higher generation rates, however, the delay curves converge, suggesting diminishing benefits of parallelization under heavy load.

Figure 5b presents the number of dropped messages as the packet generation rate per device increases. For all configurations, packet drops remain negligible at low generation rates, indicating that the gateway can initially accommodate incoming traffic. As the system approaches saturation, the number of dropped messages grows rapidly, with earlier and steeper increases for configurations with fewer cores. In particular, the single-core gateway exhibits the highest number of dropped messages across the evaluated range, while configurations with additional cores delay and significantly reduce packet loss.

Figure 5c shows the gateway throughput as a function of the packet generation rate per device. For all configurations, throughput initially increases with the offered load, reflecting effective utilization of processing resources. Each configuration eventually reaches a saturation point beyond which throughput remains approximately constant. Importantly, this saturation level increases with the number of gateway cores: while the single-core gateway stabilizes at a low throughput, configurations with two, three, four, and five cores reach progressively higher throughput plateaus. Beyond saturation, further increases in generation rate do not improve throughput, but instead result in higher delays and increased packet loss, consistent with the trends observed in Figures 5a and 5b.

This case study demonstrates that increasing the number of gateway cores significantly improves system scalability by delaying saturation and increasing the maximum achievable throughput. Higher processing capacity allows the system to sustain higher packet generation rates per sensor with lower delays and fewer packet drops, highlighting the central role of gateway dimensioning in centralized IoT architectures.

6.3. Implications for Gateway Dimensioning and Planning

From a practical perspective, the observed saturation plateaus define the operational boundary of the gateway, indicating that when aggregate traffic exceeds the approximately 6.5 packets per second threshold, further increases in sensor density result in a nonlinear degradation of system performance, particularly end-to-end latency.

These findings suggest that sensor admission control is a critical factor in network planning, as the sharp transitions in end-to-end time demonstrate that adding devices is not a linear process and requires a precise understanding of the tipping point where queuing delay becomes prohibitive for real-time applications. Furthermore, while Scenario 2 shows that horizontal scaling through additional processing cores effectively shifts the saturation point, the convergence of delay curves under heavy load implies diminishing returns if the input rate is not regulated, since the physical queue capacity itself eventually becomes the primary bottleneck. Ultimately, IoT-TSim can serve as a decision-support tool to help explore hardware dimensioning alternatives, such as the balance between processing cores and buffer capacity.

7. Conclusion

This paper introduced IoT-TSim, a lightweight, time-oriented discrete-event simulator for analyzing message traffic behavior in gateway-centric IoT architectures. The simulator models IoT devices as stochastic message sources and the gateway as a finite-capacity system with parallel processing, enabling the evaluation of end-to-end time, throughput, and dropped messages under different traffic and capacity configurations. The results show that increasing the number of sensors accelerates gateway saturation, leading to sharp increases in time and dropped messages, while gateway throughput remains bounded by its service capacity. The conducted case studies further demonstrate the feasibility of using the simulator as a quantitative support tool for gateway capacity planning, allowing the assessment of admissible message rates per device, the maximum number of supported devices, and the processing capacity required to sustain target performance levels. In contrast, increasing gateway parallelism delays saturation and improves performance, though with diminishing returns at high loads. Analytical validation based on an SPN model demonstrated strong agreement with the simulation results, with no statistically significant differences in throughput across all evaluated scenarios.

Future work includes incorporating richer temporal traffic patterns and adaptive gateway control mechanisms, enabling the investigation of scalability, load balancing, and congestion mitigation strategies in large-scale IoT systems.

Acknowledgements

The authors acknowledge the partial support of the *National Council for Scientific and Technological Development (CNPq)*, Brazil, through project number 401147/2025-8.

References

Alavi Fazel, I. and Wainer, G. (2024). Discrete event system specification for iot applications. *Sensors*, 24(23):7784.

Araujo, I., Barbosa, V., Lima, L. N., Silva, L. G., Brito, C., Fé, I., Lopes, L., Andrade, E., Leao, E., and Silva, F. A. (2025). Assessing the performance of a fault tolerant lorawan architecture with a focus on the sensor layer and data retransmission strategy. *Cluster Computing*, 28(4):275.

Bounceur, A., Marc, O., Lounis, M., Soler, J., Clavier, L., Combeau, P., Vauzelle, R., Lagadec, L., Euler, R., Bezoui, M., and Manzoni, P. (2018). Cupcarbon-lab: An iot emulator. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2.

Cantero, F. D., Corchero, J. Á. B., Toledano, M. Á. P., and Clemente, P. J. (2025). A simulation framework for assessing and optimizing iot service and resource allocation: Simulateiot-services. *Internet of Things*, page 101736.

Dauda, A., Flauzac, O., and Nolot, F. (2023). Iot: A universal dynamic gateway. In *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6.

Dayalan, U. K., Fezeu, R. A., Salo, T. J., and Zhang, Z.-L. (2022). Kaala: scalable, end-to-end, iot system simulator. In *Proceedings of the ACM SIGCOMM Workshop on Networked Sensing Systems for a Sustainable Society*, pages 33–38.

Elgazzar, K., Khalil, H., Alghamdi, T., Badr, A., Abdelkader, G., Elewah, A., and Buyya, R. (2022). Revisiting the internet of things: New trends, opportunities and grand challenges. *Frontiers in the Internet of Things*, 1.

Firouzabadi, A. D., Mellah, H., Manzanilla-Salazar, O., Khalvandi, R., Therrien, V., Boutin, V., and Sansò, B. (2023). Piot: A performance iot simulation system for a large-scale city-wide assessment. *IEEE Access*, 11:56273–56286.

Lera, I., Guerrero, C., and Juiz, C. (2019). Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758.

Maciel, P., Matos, R., Silva, B., Figueiredo, J., Oliveira, D., Fé, I., Maciel, R., and Dantas, J. (2017). 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)*, pages 50–57. IEEE.

Maciel, P. R. M. (2023). *Performance, reliability, and availability evaluation of computational systems, volume I: performance and background*. Chapman and Hall/CRC.

Mahmud, R., Pallewatta, S., Goudarzi, M., and Buyya, R. (2022). Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, 190:111351.

Mechalikh, C., Taktak, H., and Moussa, F. (2021). Pureedgesim: A simulation framework for performance evaluation of cloud, edge and mist computing environments. *Computer Science and Information Systems*, 18(1):43–66.

Nawab, F. and Shadmon, M. (2024). The tipping point of edge-cloud data management. CIDR.

Sonmez, C., Ozgovde, A., and Ersoy, C. (2018). Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493.

Statista Research Department (2026). Number of internet of things (iot) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2034. Accessed: Jan. 2026.

Ta, D.-T., Khawam, K., Lahoud, S., Adjih, C., and Martin, S. (2019). Lora-mab: Toward an intelligent resource allocation approach for lorawan. In *2019 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE.

Zeng, X., Garg, S. K., Strazdins, P., Jayaraman, P. P., Georgakopoulos, D., and Ranjan, R. (2017). Iotsim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107.