Computational Capacity Planning for Running Multiplayer Games on Cloud and Fog Infrastructures

Melissa Alves [1], Jose Wanderlei[1], Vandirleya Barbosa[1]
Arthur Sabino[1], Luiz Nelson Lima[1], Leonel Feitosa[1]
Esteban Clua [2], Francisco Airton Silva[1]

[1]PASID Lab, Universidade Federal de Piauí (UFPI)

[2]Universidade Federal Fluminense (UFF)

{melissaalves,jose.rocha,vandirleya.barbosa}@ufpi.edu.br

{arthursabino,luizznelson,leonelfeitosa,faps}@ufpi.edu.br

{esteban@ic.uff.br}

Abstract. Introduction*: Data traffic in multiplayer games consists of real-time exchanges of commands between servers and players' devices, which demand an immediate response. As the data volume increases, the system becomes overloaded, resulting in high latency and packet loss.* ***Objective****: This work proposes a new* Stochastic Petri Net *(SPN) model to evaluate the performance of this traffic in Fog–Cloud architectures designed for large-scale multiplayer games.* ***Methodology****: We built a SPN model and validated a simplified version; the model computes utilization, mean response time, drop probability, and throughput, allowing us to analyze the impact of different capacity configurations and load-distribution strategies.* ***Results****: The results reveal a direct relationship among processing capacity, load balancing, and quality of service: increasing the number of cloud GPUs reduces latency and packet loss while boosting throughput.*
Keywords *Stochastic Petri Nets, Multiplayer Games, Fog–Cloud Computing, Performance Modeling, Capacity Planning.*

1. Introduction

The global number of gamers is estimated to reach 3.42 billion in 2024, representing an annual increase of 4.5%, primarily driven by the growth of the PC gaming segment [Newzoo 2024]. This growth is reflected in the impact of electronic gaming on global internet traffic, which now accounts for 8% of all downstream data, alongside video streaming and web browsing, which together make up more than three-quarters of all traffic [Incorporated 2019]. The increasing volume of gaming traffic highlights the technical challenges of managing these demands and their importance as a reference for optimizing network systems.

To ensure a smooth experience, the infrastructure must meet performance and stability demands. Bare metal servers, recognized for their reliability and ability to handle intensive loads [Cheng et al. 2022], play a crucial role in supporting this. Additionally, Amazon Web Services (AWS) EC2 (Elastic Compute Cloud) instances are widely used

due to their flexibility and on-demand scalability [Malta et al. 2019]. The combination of these solutions with cloud and fog computing layers has proven effective in handling the complexity of real-time traffic and the variable demands of gamers.

Evaluating and optimizing the performance of these systems is a challenging task, traditionally limited by high costs and logistical complexities, such as player synchronization and control of external variables [Ahmed et al. 2022]. The continuous growth of the number of players exacerbates these challenges, generating overload, high latency, and packet loss, compromising the user experience [Wong et al. 2021]. Furthermore, the need for robust infrastructure and real-time data collection makes it difficult to implement fast and effective solutions.

This paper proposes an Stochastic Petri Nets (SPN) model to evaluate the performance of multiplayer game systems in Fog and Cloud architectures. SPN modeling was chosen for its ability to represent systems with concurrency and synchronization, essential features in multiplayer games, where multiple requests are processed simultaneously [Saldana et al. 2012]. Integrating this modeling with the analysis of metrics such as mean response time (MRT), drop probability (DP), throughput, and cloud layer utilization allows for improved load distribution and performance adjustments under different traffic and processing conditions. The model was experimentally validated, demonstrating accuracy in simulating multiplayer systems and providing support for optimizing resources based on Fog and Cloud infrastructures.

The remainder of this paper is organized as follows: Section 2 discusses related work and the contributions of this study. Section 3 details the proposed architecture for the model. Section 4 introduces the SPN models developed to simulate and evaluate the performance of multiplayer game traffic. Finally, Section 5 presents the conclusion of the work.

2. Related Work

This section reviews the literature on performance evaluation in multiplayer games. The study in [Moon 2024] presents an analysis of network traffic in two mobile games of distinct genres, using real packet measurements to investigate patterns such as size, arrival times, and self-similarity. The research in [Hains et al. 2020] proposes to evaluate the performance of the Gamers Private Network (GPN®) from WTFast, using thousands of traceroutes and Markov Chains modeling to compare latency, jitter, and spikes in GPN and non-GPN networks. In [Rossi et al. 2024], the authors develop objective QoE models for FPS games in the cloud accessed via mobile networks, based on subjective tests with 31 users playing CS:GO under different network conditions.

In [Benamer et al. 2024], a Genetic Algorithm-based approach is introduced for positioning latency-sensitive FPS game servers in the Fog, focusing on cost reduction and latency. The study conducted by [Ahmed et al. 2022] investigates netcode, latency, and packet loss in online FPS games, proposing a practical approach that allows players to identify network issues using accessible tools. Finally, [Olliaro et al. 2024] presents a stochastic model to evaluate the probability of success in online games distributed in a Gaming as a Service (GaaS) architecture with Virtual Network Functions (VNFs) at the edge, considering delays, losses, and player allocation.

This work differentiates itself by using SPN modeling to represent, in a

parameterizable way, the data traffic in multiplayer games, allowing for the simulation of different scenarios without relying on complex data collection or specific physical infrastructure. Although approaches such as stochastic queuing networks and Markov chains also allow for performance analysis, SPNs provide greater granularity in representing concurrent events, with the ability to model simultaneity and synchronization of commands in distributed environments. While the literature focuses on metrics such as latency, jitter, and packet loss through statistical analyses or simulations, our approach evaluates the system's behavior under various loads, integrating metrics such as MRT, throughput, drop probability, and resource utilization in a hybrid Cloud-Fog architecture, providing a more comprehensive view of performance. The model was experimentally validated, showing correspondence between simulated and real values, something still rarely addressed together in the studies analyzed. This combination of formal modeling, distributed architecture, and empirical validation makes this work an alternative for capacity planning and infrastructure optimization in multiplayer games.

3. System Architecture

This section presents the architecture proposed in this work (Figure 1), which is organized into layers. The data input layer corresponds to the information sent by the players, initiating the traffic flow through the different processing layers. A router transmits the collected data to the Fog Layer, which uses bare metal servers—physical machines that offer direct control over the hardware, without a virtualization layer [Kinzhalin et al. 2011]. This setup ensures a high level of customization and performance, which is essential for real-time data processing, as typically required in multiplayer games.

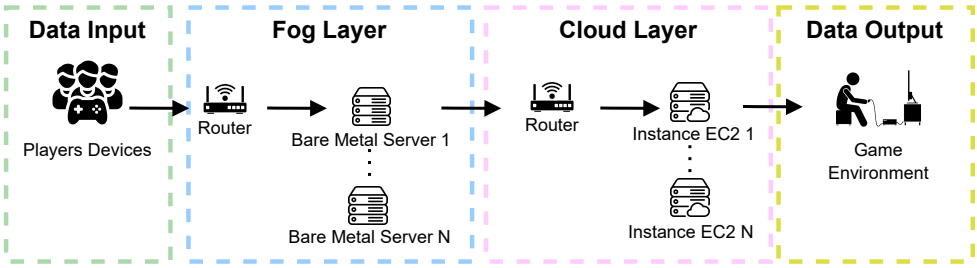


Figura 1. Multiplayer game traffic architecture.

In the Cloud Layer, the data processed by the Fog Layer is directed by a router to virtual machines running AWS EC2 instances. This elastic computing service allows for dynamic scaling of resources based on user demand [Ostermann et al. 2009]. Alternatively, other cloud infrastructure providers or bare metal solutions can be used, depending on scalability and control needs. The final layer corresponds to the forwarding of the processed data to the game environment.

Player commands are captured by devices such as controllers, keyboards, and mice, transmitted via router to the Fog Layer—a bare metal server—where they are pre-processed in real-time to ensure fast responses and continuous gameplay. This pre-processing involves the identification, classification, and validation of commands, prioritizing critical actions and discarding invalid data to minimize delays and congestion.

After this step, the commands are sent to the Cloud Layer, processed by AWS EC2 instances in a first-come, first-served manner, ensuring synchronization and integrity in the multiplayer environment, providing fast responses and a consistent experience for all players.

4. SPN Model for Performance Evaluation in Multiplayer Games

This section presents the SPN model developed based on the architecture discussed earlier. The description follows the sequence of the model's layers, as well as their described characteristics. The results obtained with the proposed model will also be presented in this section. For constructing the models and running the simulations, the Mercury tool was used [Maciel et al. 2017]. The model, illustrated in Figure 2, was developed to assist game developers in choosing the appropriate infrastructure to host their products, considering variations and changes in the capacities and configurations of the resources used.

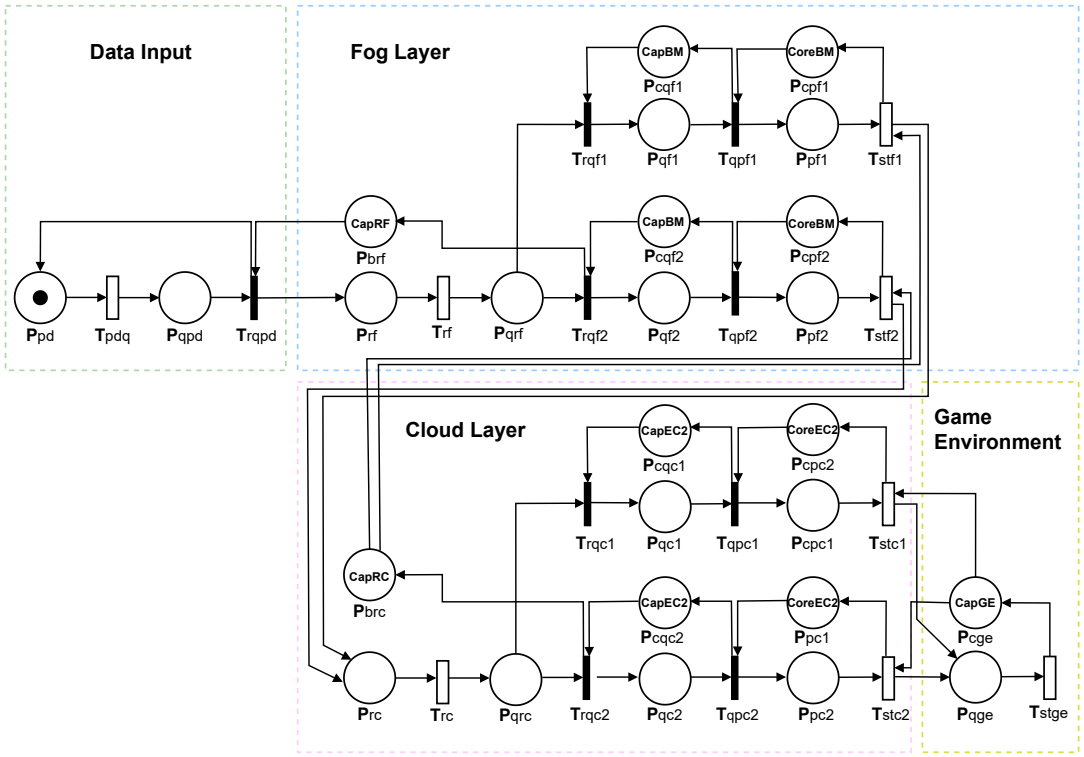


Figura 2. SPN Model for performance evaluation in multiplayer games.

The model consists of four layers: Data Input, Fog Layer, Cloud Layer, and Game Environment. In the model, the token represents a command that traverses multiple stages from its entry into the system to its application in the game. Player commands enter the system through `Ppd`, wait in the `Pqpd` queue, and are directed to the Fog Layer by the `Trqpd` transition. In the Fog Layer, the commands pass through the router's buffer (`Pbrf`), which forwards them to the `Pqrf` queue according to the time defined by the `Trf` transition. The queue's capacity is represented by the marker `CapRF`, which limits the volume of data that can be stored. The commands wait in the `Pqf1` and `Pqf2` queues, whose maximum capacity is defined by the markers `Pcqf1` and `Pcqf2`, both linked to

the marker `CapBM`. Processing occurs in the `Ppf1` and `Ppf2` queues, whose capacity is linked to the marker `CoreBM`, representing the processing capacity of the Fog Layer. Data is processed immediately as soon as capacity becomes available, without delays or associated time.

The Cloud Layer follows a pattern similar to the Fog Layer, with queues for temporary storage and processing. After processing, the commands are sent to the Game Environment, where they wait in the `Pqge` queue before being applied to the game. The capacity for updating the environment is modeled by `Pcge`, while the time needed to apply the commands to the game is represented by the `Tstge` transition. Once the tokens are triggered by `Tstge`, the capacity marked in `CapGE` is released, allowing the continuation of updates in the game environment.

4.1. Metrics

This section presents the metrics of interest used to evaluate the system's performance. The following explains how each metric was obtained using the SPN model. Equation (1) defines how the DP is calculated, indicating the probability that data will be discarded due to factors such as processing, overload, or incapacity. This metric is important for evaluating the system's ability to handle large volumes of data, avoiding the discarding of information that could alter the game state for one or both players. In the proposed model, we evaluate the probability of player commands being discarded before being sent for pre-processing, specifically at the first router. When a command is discarded, the game environment is not updated with the corresponding actions, creating a biased scenario where players with efficiently processed commands gain an advantage over those whose commands are discarded. The calculation of DP considers the probability of having tokens in a specific place. In the context of our model, this involves evaluating the drop probability at the Fog Layer router, where the absence of transfers to pre-processing indicates an issue at the system's entry point.

$$DP = P(\text{Pbrf=0}) \times 100 \tag{1}$$

This work uses Little's Law, a theorem in queuing theory, to derive the Mean Response Time (MRT) and relate the average number of tasks in a system to the average time in the system. MRT is a metric that indicates the average time required for each command to be processed and responded to by the system, allowing us to evaluate its efficiency in terms of response time for updates in the game environment. To calculate MRT, the arrival rate (AR) is defined as the inverse of the arrival delay (AD) [Little 1961], that is, AR = $\frac{1}{AD}$. The MRT calculation is based on the sum of the expectations of token existence in places related to tasks, assuming that these tokens represent queues or processing stages. This value is multiplied by AD, and the final result is obtained by dividing by $1 - DP$. Equation (2) shows how MRT is calculated.

$$MRT = \frac{\begin{array}{c}(Esp(Pqf1) + Esp(Pqf2) + Esp(Ppf1)) + (Esp(Ppf2) + Esp(Prf) + Esp(Prc)) \\ + (Esp(Pqc1) + Esp(Ppc1) + Esp(Pqc2)) + (Esp(Ppc2) + Esp(Pqge)) \times AD\end{array}}{1 - P(Pbrf = 0)} \tag{2}$$

Utilization refers to the proportion of time during which available resources are actively in use. This metric evaluates the load balancing between layers and identifies

potential bottlenecks or periods of idleness in the system. In the model, it is possible to identify utilization in specific parts of the system, providing insights into equipment configurations for the studied scenarios. This work focuses on the utilization of the Cloud Layer, the system's main layer. The calculation, shown in Equation (3), is based on the expected number of tokens in the cloud processing queue, divided by the cloud's processing capacity.

$$UC = \left(\frac{Esp(\text{Ppc1})}{\text{CoreEC2}} \right) \times 100 \tag{3}$$

Throughput indicates the number of data packets successfully processed by the system in a given time period. This metric is important for evaluating the system's processing capacity and its efficiency in handling data and transferring information. In the model, throughput is estimated based on the expected number of tokens in the place representing the game environment queue, divided by the update time required for these tokens to leave that place, as shown in Equation (4).

$$TP = \frac{Esp(\text{Pqge})}{\text{Tstge}} \tag{4}$$

4.2. Model Validation

This section presents the validation of the proposed model. The model presented in Section 4 was simplified to facilitate the validation process. To do this, some features were disregarded, such as the parallelism generated by the presence of multiple nodes in each layer, as this functionality can be omitted in the model validation. Additionally, the transitions `Tpdq` and `Trqpd` were altered from an exponential time distribution to a deterministic one. Thus, the firing of these transitions has a fixed value, unaffected by small variations caused by the exponential time distribution. Both changes were made to avoid fluctuations in the collected MRT, while still maintaining the characteristics of an SPN model. Figure 3 presents the SPN model with the aforementioned simplifications. In the experiment, the processing performed by the threads simulates the real behavior of bare metal servers in the Fog Layer and GPUs in the Cloud Layer. By running the threads, we are practically replicating the processing step that would occur in these physical components, ensuring that the model captures the essential characteristics of the distributed system.

In the validation, three layers are included: Data Input, Fog, and Cloud. Data transfer between the system components is carried out via sockets. The Data Input layer is responsible for generating the data that will traverse the system until it is processed by the last layer. The generated data is sent to the Fog Layer, where it is received by the first load balancer. Its function is to redirect the received data to be processed by a non-initialized thread. Once a thread in the Fog Layer finishes processing, the corresponding data is sent to the Cloud Layer. The data distribution follows the same steps performed by the Fog Layer, passing through the layer's load balancer and being distributed among the threads contained in that layer. When a thread in the Cloud Layer finishes processing a piece of data, it is sent back to the Data Input Layer. Subsequently, the time required for the data to return to the origin is computed to calculate the MRT of the experiment with the given configuration.

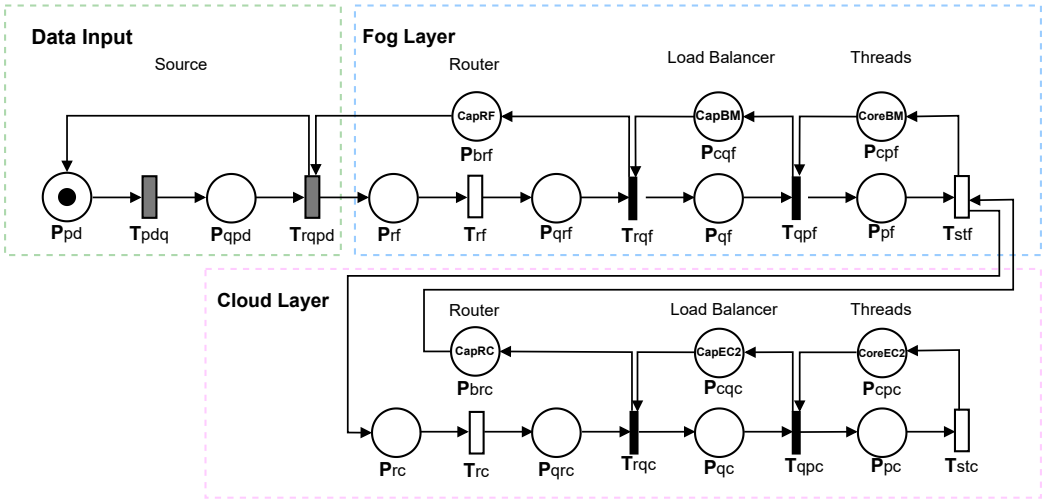


Figura 3. Simplified model for validation.

4.2.1. Experiment Environment and Analysis

The experiment was conducted in an environment with three identical machines, each with an Intel Core i3 processor at 3.7GHz and 8GB of RAM, representing different layers of the system: the first generated the data, the second acted as the fog layer, redirecting the data for processing, and the third represented the cloud. The objective was to collect the average times of each stage of the system to feed the transitions of the SPN model, using the MRT metric and the Mercury tool. The recorded times were: `Tpdq` of 200 ms, `Trqpd` of 17.10 ms, `Trf` of 23.90 ms, `Tstf` of 657 ms, another `Trf` of 25.70 ms, and `Trc` of 2105.70 ms, reflecting the system's characteristics and being essential for configuring the model.

Table 1 presents the MRT values for both the model and the experiment obtained in each evaluated configuration. In all configurations, 200 samples were used, each representing a simulation run by both the SPN model and the experiment. This table also includes the confidence interval for the MRT of the experiment, which indicates the minimum and maximum values achievable in the experiment for a given configuration. The validation of the configurations was performed using the confidence interval. If the MRT of the model falls within the confidence interval of the experimental results, we can assert that the model was validated, as this indicates that the model's predictions align with the observed data from the experiments.

Tabela 1. Response Time of the Model and the Experiment.

Cloud GPUs	Model (ms)	Experiment [Confidence Interval] (ms)
2	319766.69	260367.00 [136364.28, 384369.71]
4	107276.07	99943.00 [53790.74, 146095.25]
6	41645.06	45887.00 [25711.28, 66062.71]

4.3. Case Study

This section presents the results obtained to evaluate data traffic in multiplayer games, analyzing the impact of variations in the number of GPUs in the Cloud Layer. Simulations

were carried out considering configurations with 2, 4, 6, and 8 GPUs, aiming to understand the system's behavior under different processing configurations. The analysis focuses on metrics such as Cloud Layer utilization, MRT, DP, and throughput. Table 2 details the configuration parameters used in the model for the simulations.

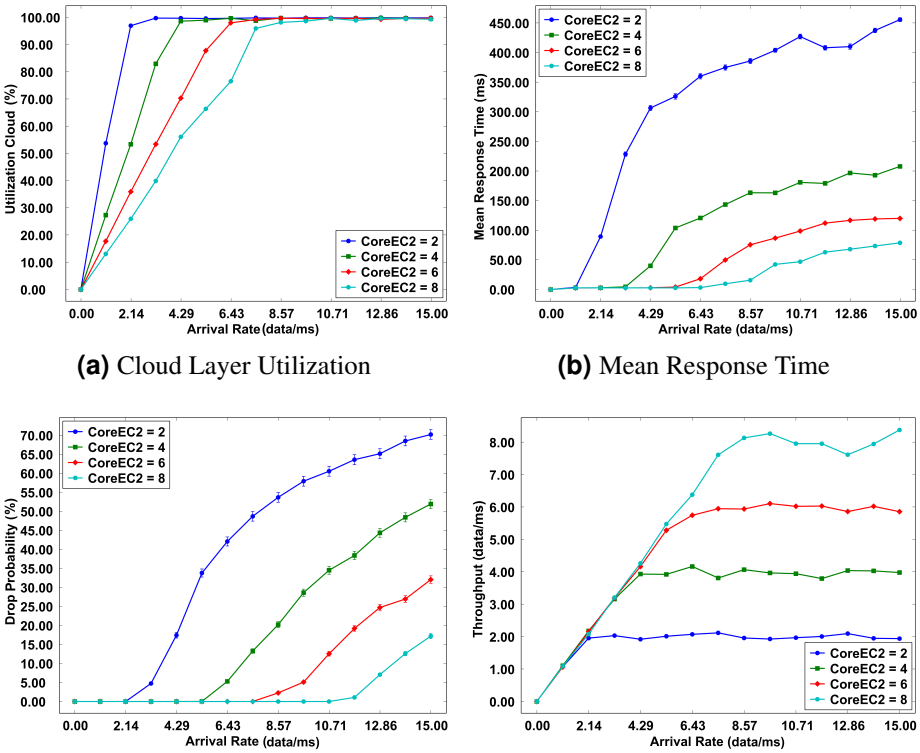
Tabela 2. Model configuration parameters.

Type	Component	Value
Timed transitions	AD	200 ms
	Trf	100 ms
	Trc	80 ms
	Tpdq	AD
	Tstf1 e Tstf2	500 ms
	Tstc1 e Tstc2	2000 ms
	Tstge	50 ms
Marked places	CapRF	100
	CapRC	1000
	CapBM	200
	CapEC2	300
	CoreBM e CoreEC2	4
	CapGE	50

The results obtained, presented in Figure 4, demonstrate the relationship between the number of GPUs and system performance. With 2 GPUs, the Cloud Layer reaches 100% utilization at an AR higher than 2.14 data/ms, leading to a sharp increase in MRT, which rises from 50 ms to 450 ms as the AR reaches 15 data/ms. As a result of the Cloud Layer overload, the DP increases drastically, reaching 70% at the highest AR. The data throughput stabilizes at around 2.14 data/ms, indicating that the system quickly reaches its maximum processing capacity and is unable to handle higher arrival rates.

With 4 GPUs, the Cloud Layer utilization reaches 100% at an AR of 4.29 data/ms, resulting in a more gradual increase in MRT compared to the previous configuration, reaching about 200 ms at the highest AR. The DP is also lower, reaching approximately 50% at the highest AR. The throughput stabilizes at 4.0 data/ms after the AR of 4.29 data/ms, suggesting that the system still faces processing limitations, though less severe than with 2 GPUs. With 6 GPUs, the Cloud Layer utilization reaches 100% only at AR$>$6.43 data/ms, while the MRT stabilizes around 100 ms, indicating higher processing efficiency. The DP decreases to approximately 30% at the highest AR, and the throughput reaches about 6.0 data/ms, demonstrating that the system can handle a higher load due to the increase in the number of GPUs. As expected, with 8 GPUs, the Cloud Layer shows the best performance in all aspects. Utilization reaches 100% only with an AR of 10.71 data/ms. The MRT remains below 50 ms until this point, rising to about 100 ms at the highest AR, while the DP decreases further to about 20%. Additionally, the throughput peaks at 8.0 data/ms, indicating that the system can handle high workloads more efficiently with a higher number of GPUs in the Cloud Layer.

The results presented in Figure 4 show a clear correlation between the increase in the number of GPUs in the Cloud Layer and improved system performance. In the configuration with 2 GPUs, the Cloud Layer overload occurs quickly, resulting in a sharp increase in MRT and DP. As the number of GPUs increases, the system supports higher ARs before reaching overload, leading to improved performance. The configuration with 8 GPUs offers the best balance between utilization, MRT, DP, and throughput.



(a) Cloud Layer Utilization **(b)** Mean Response Time

(c) Drop Probability **(d)** Throughput

Figura 4. Case study results.

5. Conclusion

This work proposes an SPN model to evaluate the performance of data traffic in multiplayer games based on fog and cloud computing. The results indicate that the number of GPUs available in the cloud layer significantly impacts the system's performance, improving scalability, allowing for higher data arrival rates, and keeping MRT and DP reduced. These findings suggest that cloud providers and game developers can use dynamic GPU scaling to optimize performance during periods of high demand. The model validation showed convergence between the experimental and modeled MRT values, confirming its effectiveness in scenarios with variations in resource availability and providing a practical tool for infrastructure planning. However, limitations such as the simplification of the model and the absence of factors like network latency and prioritization of critical commands indicate the need to explore additional metrics and analyze different types of games and workloads in future research.

Referências

Ahmed, M., Reno, S., Rahman, M. R., e Rifat, S. H. (2022). Analysis of netcode, latency, and packet-loss in online multiplayer games. In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 1198–1202. IEEE.

Benamer, A.-R., Hadj-Alouane, N. B., Boussetta, K., e Hadj-Alouane, A. B. (2024). A genetic algorithm-based approach for placement in the fog of latency-sensitive multiplayer game servers. *Cluster Computing*.

Cheng, D., Verma, R., Rane, D., Jha, R. S., e Ibrahim, W. (2022). Next-generation optimization models and algorithms in cloud and fog computing virtualization security: The present state and future. *Scientific Programming*, 2022:2419291.

Hains, G., Mazur, C., Ayers, J., Humphrey, J., Khmelevsky, Y., e Sutherland, T. (2020). The wtfast's gamers private network (gpn ®) performance evaluation results. In *2020 IEEE International Systems Conference (SysCon)*, pages 1–6.

Incorporated, S. (2019). The global internet phenomena report: Q3 2019. Accessed: November 23, 2024.

Kinzhalin, A., Kohn, R., Lombard, D., e Morin, R. (2011). Enabling dynamic data centers with a smart bare-metal server platform. *Cluster Computing*, 14(3):245–258.

Little, J. D. C. (1961). A proof for the queuing formula. *Operations Research*, 9(3):383–387.

Maciel, P., Matos, R., Silva, B., Figueiredo, J., Oliveira, D., Fé, I., Maciel, R., e 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.

Malta, E. M., Avila, S., e Borin, E. (2019). Exploring the cost-benefit of aws ec2 gpu instances for deep learning applications. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, pages 21–29. IEEE.

Moon, D. (2024). Network traffic characteristics and analysis in recent mobile games. *Applied Sciences*, 14(4).

Newzoo (2024). Global games market report 2024. Accessed: November 21, 2024.

Olliaro, D., Mancuso, V., Castagno, P., Sereno, M., e Marsan, M. A. (2024). Gaming on the edge: Performance issues of distributed online gaming. In *2024 IFIP Networking Conference (IFIP Networking)*, pages 259–267.

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., e Epema, D. (2009). A performance analysis of ec2 cloud computing services for scientific computing. In *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 115–131. Springer.

Rossi, H. S., Mitra, K., Åhlund, C., Cotanis, I., Örgen, N., e Johansson, P. (2024). Objective qoe models for cloud-based first person shooter game over mobile networks. In *2024 IEEE 21st Consumer Communications and Networking Conference (CCNC)*, pages 550–553.

Saldana, J., Suznjevic, M., Sequeira, L., Fernandez-Navajas, J., Matijasevic, M., e Ruiz-Mas, J. (2012). The effect of tcp variants on the coexistence of mmorpg and best-effort traffic. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, page 1–5. IEEE.

Wong, A., Chiu, C., Hains, G., Behnke, J., Khmelevsky, Y., e Mazur, C. (2021). Modelling network latency and online video gamers' satisfaction with machine learning. In *2021 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, pages 1–5.